UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Automática y Electrónica Industrial

CONTROLADOR DE UN HORNO DE REFLUJO DE PRECISIÓN PARA SOLDADURA DE COMPONENTES SMT

Memoria y Anexos

Autor: Javier Rodrigo Aranda
Director: Alfonso Conesa Roca

Convocatoria: Mayo 2019

Resumen

En el presente documento se describe el diseño e implementación de un controlador para un horno de soldadura por reflujo para componentes electrónicos de montaje superficial a partir de un horno convencional de sobremesa. Dicho controlador ha de regular la temperatura del horno a los perfiles de soldadura establecidos en la industria de forma adecuada y fiable.

Gracias a los dispositivos y la alta conectividad que existen a día de hoy se decidió dotar al controlador con conectividad Wi-Fi, este es capaz de conectarse a nuestra red local o crear un punto de acceso Wi-Fi. Mediante esta conectividad el controlador es capaz de ofrecer una interfaz web, accesible desde cualquier navegador web, en la cual se pueden visualizar los diferentes perfiles de soldadura, una gráfica en tiempo real del proceso y editar las diferentes opciones con las que cuenta.

Se decidió también añadirle una interfaz de control mediante una pantalla de cristal líquido y un teclado táctil capacitivo, para así ofrecer un control más directo sin necesidad de estar conectado al dispositivo.

Finalmente el controlador se ha concebido como dos productos utilizando los mismos componentes de base. La primera variante se ha concebido para los usuarios que deciden montárselo por su cuenta, por ello se ha integrado todo lo necesario para su funcionamiento en el circuito de control, pudiendo así ofrecer solo la placa de control como un solo producto. La segunda variante o producto final sería el controlador totalmente ensamblado y listo para su uso.

Resum

En el present document es descriu el disseny i implementació d'un controlador per a un forn de soldadura per reflux per a components electrònics de muntatge superficial a partir d'un forn convencional de sobretaula. Aquest controlador ha de regular la temperatura del forn segons els perfils de soldadura establerts en la indústria de forma adequada i fiable.

Gràcies als dispositius i l'alta connectivitat que existeixen avui dia es va decidir dotar al controlador amb connectivitat Wi-Fi, aquest és capaç de connectar-se a la nostra xarxa local o crear un punt d'accés Wi-Fi. Mitjançant aquesta connectivitat el controlador és capaç d'oferir una interfície web, accessible des de qualsevol navegador web, en la qual es poden visualitzar els diferents perfils de soldadura, una gràfica en temps real del procés i editar les diferents opcions amb les quals compte.

Es va decidir també afegir-li una interfície de control mitjançant una pantalla de cristall líquid i un teclat tàctil capacitiu, per a així oferir un control més directe sense necessitat d'estar connectat al dispositiu.

Finalment el controlador s'ha concebut com dos productes utilitzant els mateixos components de base. La primera variant s'ha concebut per als usuaris que decideixen muntar-lo pel seu compte, per això s'ha integrat tot el necessari per al seu funcionament en el circuit de control, podent així oferir només la placa de control. La segona variant o producte final seria el controlador totalment assemblat i llest per al seu ús.

Abstract

This document describes the design and implementation of a reflow controller for soldering electronic surface mount components using a conventional table top oven. The controller must regulate the temperature of the oven based on soldering profiles in an adequate and reliable way.

Thanks to the devices and their higher connectivity that exists today, I was decided to add Wi-Fi connectivity to the reflow controller, it's able to connect to our local network or create a Wi-Fi hotspot. Through this connectivity the reflow controller is able to provide a web interface, accessible from any web browser, in which the different soldering profiles can be visualized, a real-time graph of the current process and edit the different profile options.

I was also decided to add a control interface through a liquid crystal display with a capacitive touch keypad, to offer a more direct control without having to be connected to the device.

Finally, the controller has been conceived as two products using the same base components. The first variant has been conceived for users who decide to install it by themselves, for that reason, all the necessary components for its correct operation are included in the main control board. The second variant or final product would be the controller fully assembled and ready for use.

Glosario

CPU: Unidad central de procesamiento (del inglés *central processing unit*)

IoT: Internet de las cosas (del inglés *Internet of Things*)

IP: Dirección IP (del inglés *Internet Protoco*l)

ITC-BT: Instrucción técnica complementaria – Baja Tensión

LCD: Pantalla de cristal líquido (del inglés *Liquid Crystal Display*)

NTC: Coeficiente de temperatura negativo (del inglés *Negative Temperature Coefficient*)

RMS: Valor cuadrático medio (del inglés *Root Mean Square*)

RST: Reiniciar (del inglés *Reset*)

SMD: Dispositivo de montaje superficial (del inglés S*urface-Mount Device*)

SMT: Tecnología de montaje superficial (del inglés *Surface-Mount Technology*)

SoC: Sistema en chip (del inglés D*ystem on a Chip*)

THT: Tecnología de agujeros pasantes (del inglés *Through-Hole Technology*)

VAC: Voltaje en alterna y valor eficaz

WPA: Acceso protegido Wi-Fi (del inglés *Wi-Fi Protected Access*)

Índice

Capítulo 1. Introducción

1.2. Origen del trabajo

Este trabajo nace de mi necesidad para la mejora de la producción en el ensamblado de circuitos electrónicos, ya que debido a mi trabajo actual como autónomo todo el soldado de los componentes lo hacía de manera manual. Esto se traducía en tiempos de ensamblado por placa demasiado elevados.

El primer paso para mejora la producción fue la creación de una maquina "Pick and Place" para la colocación de los componentes en la placa, que después soldaba manualmente mediante una pistola de aire caliente. Pero en este último proceso me provocaba una tasa de fallos mayor que el soldado manual, ya que en algunos casos las placas eran expuestas a una sobre temperatura debido a la imprecisión del proceso de soldado por aire caliente.

Por ello decidí incorporar un horno con control digital de temperatura y así poder reproducir siempre una misma curva de temperatura, la cual se denomina perfil de soldadura.

1.3. Objetivos del trabajo

El objetivo principal de este proyecto es la creación de un circuito electrónico que sea capaz de monitorear la temperatura de un horno convencional, para así controlar los elementos calefactores del mismo y obtener un control preciso de la temperatura.

Los objetivos secundarios de este proyecto son:

- Utilización de tecnologías actuales de conectividad Wi-Fi.
- Control mediante una interfaz web, para maximizar la compatibilidad con cualquier dispositivo, sin necesidad de una aplicación específicamente desarrollada para cada plataforma.
- Que sea capaz de competir con otros productos actualmente disponibles en el marcado.

1.4. Alcance del trabajo

El alcance de este proyecto es el desarrollo e implementación de un controlador para un horno de reflujo, tanto a nivel de hardware como a nivel de software.

La parte del hardware de control será capaz de controlar un horno convencional de sobremesa mediante la adquisición de la temperatura interna del mismo a través de un termopar. También se ha decidió diseñar una interfaz de entrada compuesta por una pantalla de cristal líquido y un teclado táctil capacitivo.

A nivel de software se implementara todo lo necesario para el correcto funcionamiento del control de temperatura, la pantalla de cristal líquido y control táctil capacitivo. Integrada en la misma programación, junto a todo lo anterior, se incluirá una interfaz web para la visualización gráfica de los diferentes perfiles de soldadura, graficar en tiempo real la curva actual de temperaturas y la modificación de la configuración de los diferentes perfiles de soldadura.

Capítulo 2. Estado del Arte

2.1. Soldadura por reflujo o refusión

La soldadura por reflujo es un proceso en el que mediante una pasta de soldadura, previamente extendida mediante una plantilla, se fijan los componentes temporalmente a las placas de circuito impreso gracias a su viscosidad, para después aplicarle calor y fijar los componentes de manera permanente.

La pasta de soldadura que se utiliza en dicho proceso es una mezcla de partículas de una determinada aleación y un fundente que elimina los óxidos para favorecer la soldadura. Las aleaciones más utilizadas en este proceso son Sn-Cu-Ag, con un punto de fusión 226 ^{o}C similar al Sn-Pb restringido su uso debido a su contenido en plomo, y Sn-Bi con un bajo punto de fusiones de 139^{o}C, este último utilizado para componentes sensibles a las altas temperaturas.

Este proceso no solo sirve para la soldadura de competentes SMT (Tecnología de montaje superficial), hay una técnica poco extendida denominada *Paste-in-Hole* la cual añade la pasta de soldar en los agujeros de los componentes del tipo THT (Tecnología de agujeros pasantes) para así ser soldados en el horno de reflujo junto a los componentes SMT. De esta manera no se requiere un equipamiento extra para la soldadura de los componentes del tipo THT.

En el proceso de soldadura mediante reflujo hay normalmente cinco fases, también llamadas zonas, cada una con un perfil térmico. Estas fases son:

- **Evaporación**: Se evaporaran los disolventes de la pasta de soldar.
- **Activación**: Se activa el fundente y se deja que actúe.
- **Precalentado**: Se precalientan los componentes y el circuito impreso.
- **Reflujo**: Se derrite la pasta de soldar que une los componentes a la placa.
- **Enfriado**: Se enfría la placa a una velocidad controlada y hasta una temperatura aceptable.

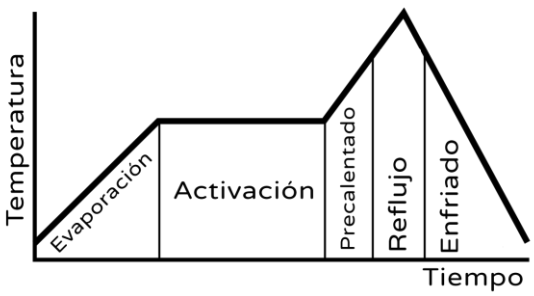


Figura 2.1. Ejemplo de un perfil de soldadura.

2.2. Estudio de mercado

En el mercado existen una gran variedad de hornos de reflujo dependiendo de la escala de producción que se requiera, desde los grandes hornos industriales de convección hasta los pequeños hornos de infrarrojos para el soldado de pequeños lotes. Pero el principal problema es el coste de los mismos, suponiendo una inversión muy grande en el caso de que la producción no sea lo suficientemente elevada.

Por ello existe una alternativa que es el uso de un horno convencional que mediante un controlador externo se regula la temperatura. Aquí es donde se encuentra el problema, ya que existen muchas alternativas pero ninguna de ellas me parece segura para el usuario final, ya que depende de él mismo el ensamblaje, o suponen un coste muy elevado.

2.3. El internet de las cosas

El internet de las cosas (del inglés "*Internet of Things*"), de aquí en adelante IoT, es un término que hace referencia a una red de objetos físicos interconectados a través de Internet. El concepto fue acuñado por Kevin Ashton del MIT en 1999.

Es un concepto muy sencillo pero de difícil aplicación, ya que si todos los objetos como libros, camisetas, paquetería o partes de un vehículo estuvieran interconectados con dispositivos de identificación, la vida rutinaria sufriría una transformación radical. Desaparecerían los objetos perdidos, sabríamos concretamente que se consume en cada momento. Al mismo tiempo los robos desaparecerían, puesto que cada producto contaría con una identificación única y sabríamos en todo momento donde se encuentra.

2.4. Concepción del producto

Para alcanzar a un mayor número de usuarios se ha planteado el producto como dos variantes. Una primera variante básica, que requiere que el usuario final la ensamble, y una totalmente ensamblada lista para su uso.

La primera variante básica incluiría solo la placa de control. Esta versión básica requiere por parte del usuario final el montaje para su funcionamiento (de manera interna o externa en el mismo horno). Por este motivo la placa de control deberá contar con todos los componentes necesarios para su funcionamiento independientemente de los periféricos utilizados y así poder ofrecer un primer producto básico pero funcional a un coste más reducido.

Para la versión totalmente ensamblada se ha optado por un montaje dentro de una caja plástica IP65 y así asegurar que no hay partes metálicas que puedan provocar una fuga eléctrica hacia el operario de la misma.

Figura 2.2. Caja para el montaje renderizada en Solidworks a partir de las dimensiones del fabricante.

En este punto se decidió añadir una pantalla de cristal líquido junto a un teclado táctil capacitivo. El motivo principal fue la necesidad de poder controlarlo de manera más local sin la necesidad de un dispositivo externo. La decisión de utilizar un teclado táctil capacitivo es la capacidad de poder leer las diferentes pulsaciones a través del plástico de la caja obteniendo así un producto con todas sus partes metálicas aisladas.

Capítulo 3. Diseño del Hardware del Controlador

En este apartado se especifica toda la implementación física del producto la cual se denominara de aquí en adelante como *hardware*.

3.1. Requisitos del diseño

El parámetro principal del diseño es la potencia máxima a controlar, para ello se han tenido en cuenta diferentes parámetros. El primero de ellos es la potencia máxima que puede soportar una toma de corriente de uso general, especificado en el "Reglamento Electrotécnico para Baja Tensión" (sección ITC-BT-25), siendo de 3450 W (16 A) para un circuito de tomas de corriente.

Otro de los parámetros tenidos en cuenta ha sido la potencia nominal de los hornos de sobremesa disponibles en el mercando, tomando una muestra de 44 modelos de los cuales 42 consumen una potencia nominal de 2000 W o menos.

De esta manera se decidió escoger como potencia nominal máxima de control de 2000 W.

3.2. Descripción de los componentes

En este apartado se describe y justifica el diseño de las diferentes partes del circuito electrónico.

3.2.1. Microcontrolador

Siendo esta la parte más importante del circuito de control y para poder cumplir con los objetivos del proyecto se ha decidido utilizar un SoC (del inglés *system on chip*). Este debe contar con conectividad Wi-Fi y memoria interna suficiente como para poder gestionar una interfaz web de control.

Por ello se ha decidido utilizar el SoC ESP32 de la compañía Espressif, ya que este cuenta con muchas más funciones de las necesarias en este proyecto haciéndolo ideal para el mismo.

Tabla 3.1. Características principales del SoC ESP32-D0WD.

CPU	Doble núcleo RISC de 32-Bit
Frecuencia	160 o 240 MHz
Memoria SRAM	512 KiB
Memoria flash externa	4 a 16 MiB
Wi-Fi	IEEE 802.11b/g/n a 2.4 GHz
Criptografía Wi-Fi	WPA/WPA2, WFA y WAPI
Bluetooth	Versión 4.2
Buses	UART, SPI, I^2C, Ethernet, CAN, IR, SDIO
Conversor analógico digital	18 canales de 12 bits

Por otra parte la compañía Espressif ofrece el SoC ya ensamblado en módulos con la antena y la memoria flash ya integradas. Estos módulos cuentan con todas las certificaciones aptas para el mercado de los diferentes países del mundo (CE de Europa, FCC de Estados Unidos, Wi-Fi, MIC de Japón, KCC de Corea del Sur, …) de esta manera se asegura el desarrollo de un producto apto para el mercado de todos los países.

En concreto se ha decidido utilizar el módulo **ESP32-WROOM**, este cuenta con 4MiB de memoria interna y la antena ya integrada en el circuito impreso. También hay disponibles otros módulos que cuentan con codificación/decodificación de audio y video, antena externa o más cantidad de memoria flash y SRAM.

Figura 3.1. Módulo ESP32-WROOM (Fuente: Espressif).

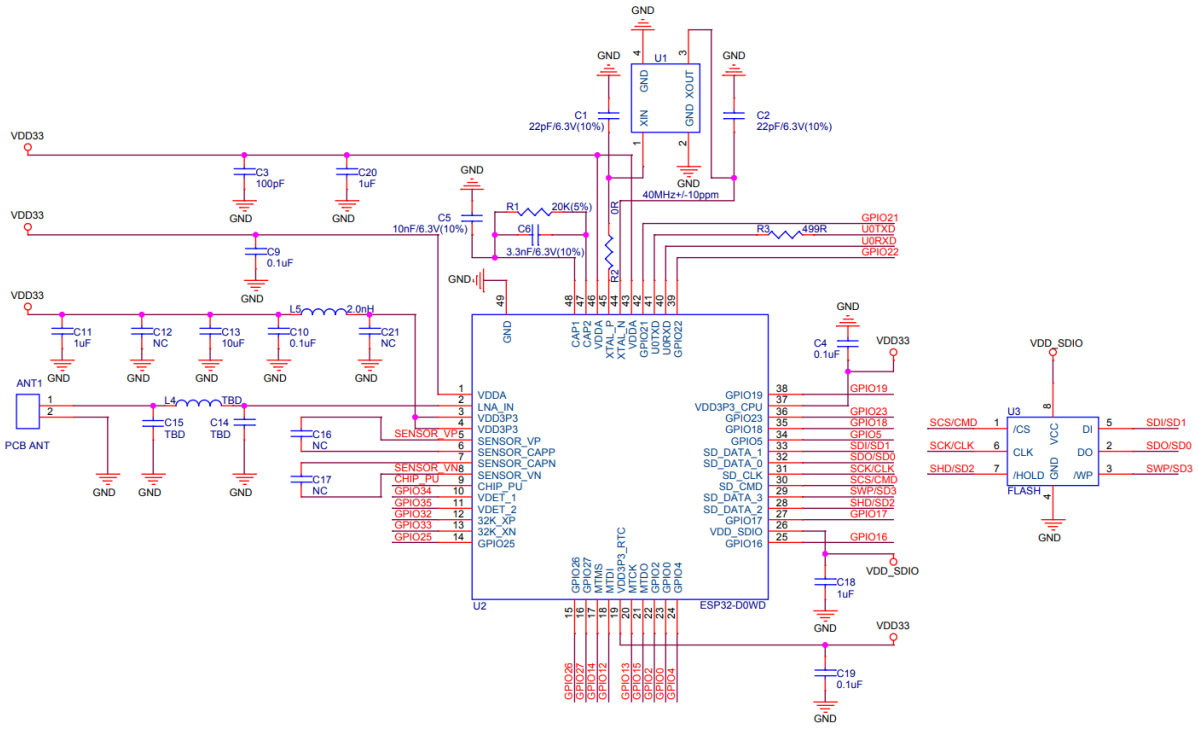


Figura 3.2. Esquemático del módulo ESP32-WROOM (Fuente: Hoja de datos del fabricante).

Para la programación de este módulo se requiere un circuito especial el cual permite entrar en el modo de programación y subir el programa mediante un adaptador de USB a puerto serie. Este circuito es el mismo que utiliza el fabricante en sus placas de desarrollo, con la diferencia que en este caso se ha decidido utilizar un módulo USB a puerto serie externo.

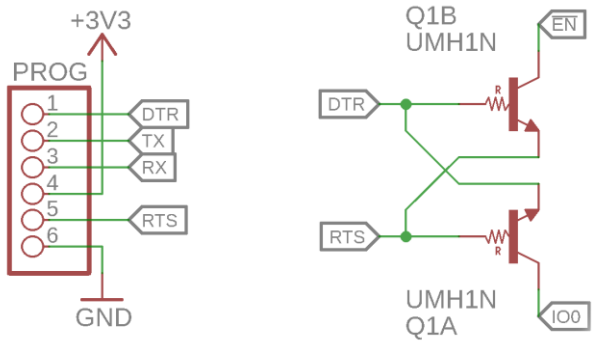


Figura 3.3. Esquemático del control de las señales y puerto de programación.

Para entrar en el modo de programación el modulo requiere que el GPIO 0 este en estado bajo antes que el pin de *ENABLE* pase a estar en estado alto. Esto se consigue mediante el transistor Q1 UMH1N del fabricante ROHM el cual integra dos transistores en un mismo encapsulado con sus respectivas resistencias de base.

Tabla 3.2. Características principales del UMH1N

Tensión de alimentación máxima	50 V
Corriente de salida máxima (Colector)	100 mA
Corriente de entrada máxima (Base)*	360 µA a 5V

*Fijada por las resistencias internas.

De esta manera y a través de las señales de control del puerto serie, se obtiene la secuencia adecuada a la hora de volcar la programación en la memorias flash del módulo.

Tabla 3.3. Lógica de funcionamiento

Señales de entrada		**Señales de salida**	
DTR	**RTS**	**ENABLE**	**GPIO 0**
1	1	1	1
0	0	1	1
1	0	0	1
0	1	1	0

Para la programación del dispositivo se requiere un adaptador USB que sea capaz de emular un puerto serie. Para ello yo he utilizado un adaptador diseñado por mi basado en el chip FT232RL, capaz de emular un puerto serie a través de un puerto USB estándar. Esta cuenta con un regulador de voltaje de 3,3 V capaz de entregar hasta 500mA de corriente para así alimentar todo el conjunto sin ningún problema. La única modificación que hay que efectuar es un puente entre las señales de CTS y RTS para poder entrar en el modo de programación.

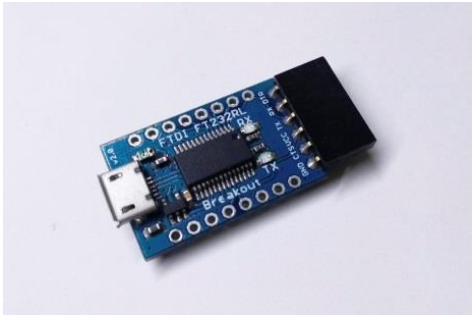


Figura 3.4. Adaptador USB a Puerto Serie.

Este adaptador fue diseñado como hardware de libre distribución, por consiguiente los archivos de diseño están disponibles a través de mi página web (jrodrigo.net).

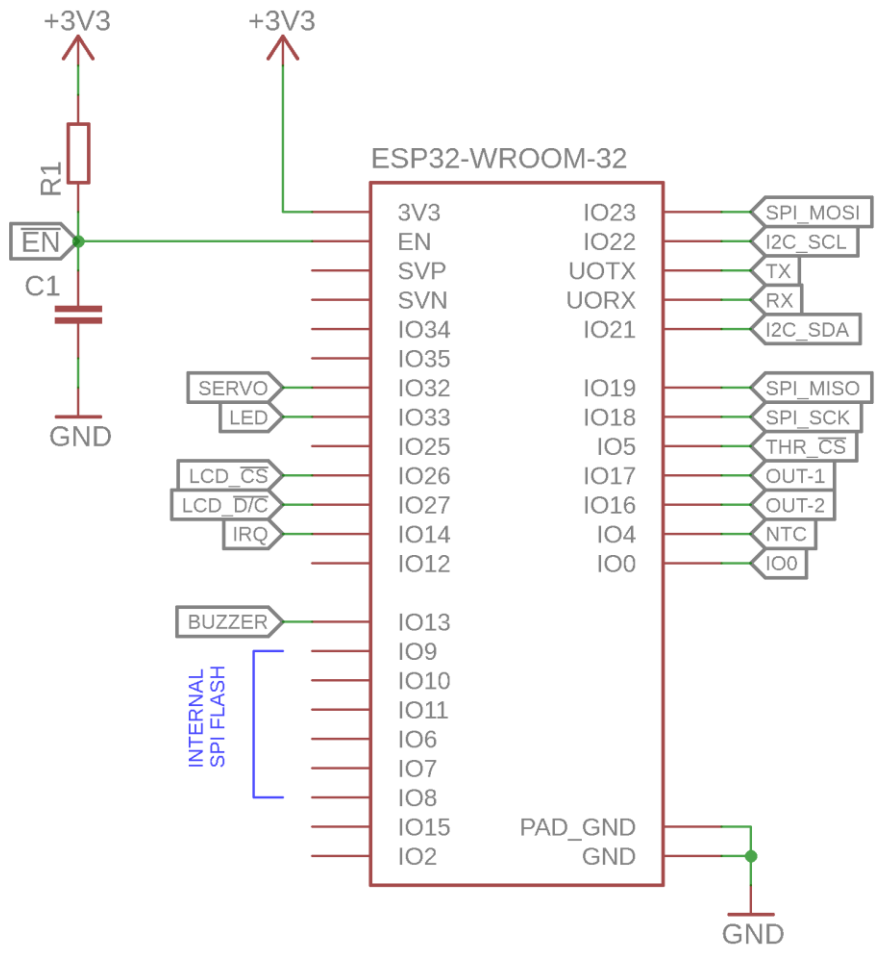


Figura 3.5. Esquemático de conexionado del módulo ESP32-WROOM.

3.2.2. Fuente de alimentación

Para la alimentación se ha optado por utilizar una fuente de alimentación conmutada integrada la cual es capaz de funcionar en redes de 120 VAC o 230 VAC indistintamente, de esta manera el controlador puede ser utilizado en cualquier país.

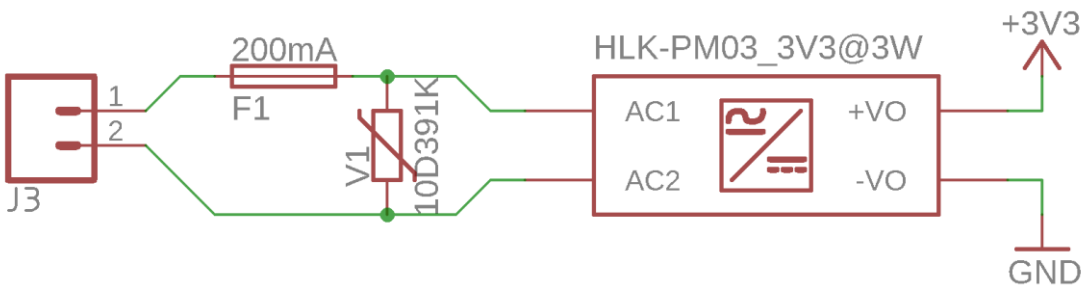


Figura 3.6. Esquemático de la fuente de la fuente de alimentación conmutada y sus protecciones.

Se ha añadido dos protecciones de entrada como seguridad en casos de sobrecarga de la fuente o sobretensiones de línea, formada por un fusible F1 del tipo cilíndrico de 200mA y un varistor V1 el cual trabaja a 250VAC. Estas protecciones actuaran en los siguientes casos:

- Cuando la fuente de alimentación sufra una sobrecarga por encima de los 200mA el fusible F1 se fundirá cortando la alimentación.
- En picos de sobretensiones de la red el varistor V1 cortocircuitara la fase y el neutro evitando que este le afecte a la fuente de alimentación.
- En caso de sobretensión prolongada, al igual que el caso anterior el varistor cortocircuitara la fase y el neutro, pero en este caso el fusible F1 se fundirá desconectando la fuente y el varistor.

Tabla 3.4. Características principales de la fuente de alimentación HLK-PM03.

Tensión de alimentación (AC)	100 – 240 V
Consumo de entrada máximo	200 mA
Tensión de salida (DC)	3,3 ± 0,1 V
Corriente máxima de salida	1 A

Tabla 3.5. Características principales del varistor 10D391K.

Max. Voltaje continuo (RMS)	250 V
Max. Voltaje continuo (DC)	320 V
Corriente máxima	25 A

3.2.3. Lectura de la temperatura a través de un termopar

Se ha decidido utilizar el MAX6675 de *Maxin Integrated* para la lectura del termopar del tipo K (Níquel-Cromo/Níquel-Aluminio), el cual realiza la compensación de la unión fría internamente, digitaliza la medida de la unión caliente con una resolución de 12 Bits o 0,25 °C y puede medir hasta 1024 °C.

Tabla 3.6. Características principales del MAX6675.

Tensión de alimentación	3,0 – 5,5 V
Consumo típico	0,7 mA
Resolución	12 bit y 0,25 ^{o}C
Rango de medida	0 – 1023,75 ^{o}C
Termopar del tipo	K

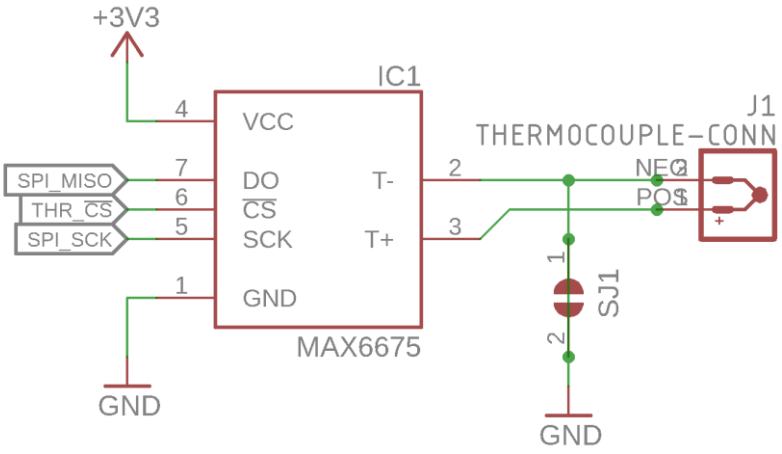


Figura 3.7. Esquemático de conexionado del MAX6675.

3.2.4. Protección de sobre temperatura

Para evitar que el circuito pueda usarse en estado de sobre temperatura se ha colocado un termistor en el circuito impreso. Este se ha colocado junto a la fuente de alimentación ya que es el componente más sensible a las altas temperaturas debido a los condensadores electrolíticos que utiliza internamente.

El termistor es del tipo NTC (Coeficiente de temperatura negativo), escogiendo el NTCLE100E3 del fabricante Vishay. El circuito también cuenta con una resistencia secundaria formando un divisor resistivo para la medida de la temperatura. Para obtener una medida más estable en el tiempo se ha añadido un condensador y así también poder evitar ruido indeseado.

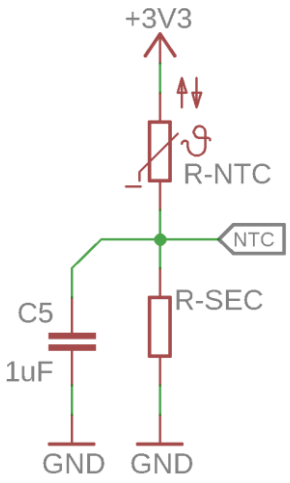


Figura 3.8. Esquemático de medida del termistor NTC NTCLE100E3.

El termistor es un transductor del tipo resistivo, por ese motivo se ha montado en un divisor resistivo creando una variación de voltaje en función de la temperatura del mismo. La resistencia del termistor varía de manera exponencial en función de varios parámetros que nos proporciona el fabricante.

Tabla 3.7. Características principales del termistor NTCLE100E3.

T_0	25 ºC
Resistencia a 25 ºC	10 KΩ ± 5%
$B_{85/25}$	3977 ± 5%
Rango de temperatura	-40 a +125 ºC

Mediante los parámetros que nos proporciona el fabricante y la ecuación 3.1 se ha obtenido una gráfica que nos muestra la evolución de la resistencia en función de la temperatura.

$$R_{NTC} = R_0 \cdot e^{\beta\left(\frac{1}{T}-\frac{1}{T_0}\right)} \qquad \textbf{(Eq. 3.1)}$$

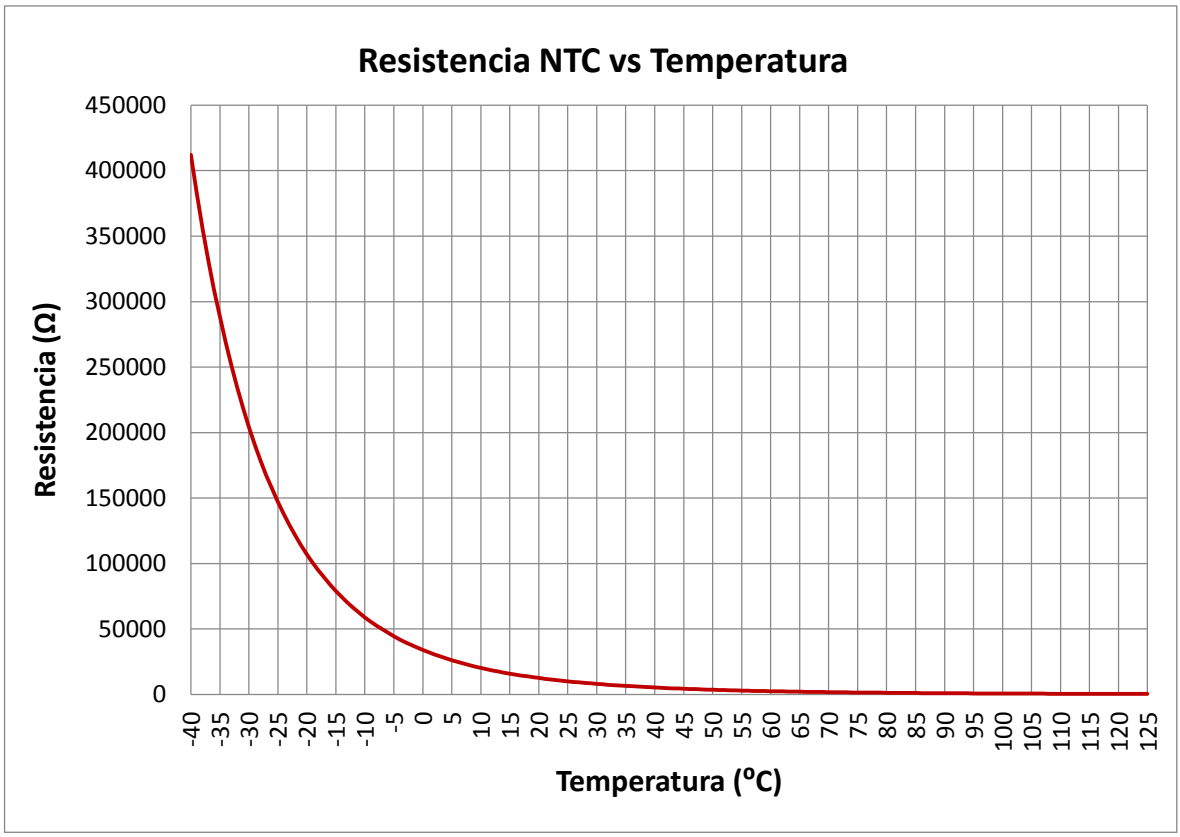


Figura 3.9. Gráfica de la resistencia del termistor en función de la temperatura.

En la gráfica anterior podemos observar la resistencia del termistor en todo el rango de temperatura. Pero para un sensado óptimo de la temperatura se ha escogido el rango de 50 a 100 grados centígrados (figura 3.10). En ese rango es donde se encuentra el punto donde desactivaremos el circuito o/y se emitirá una señal acústica avisando de la sobre temperatura.

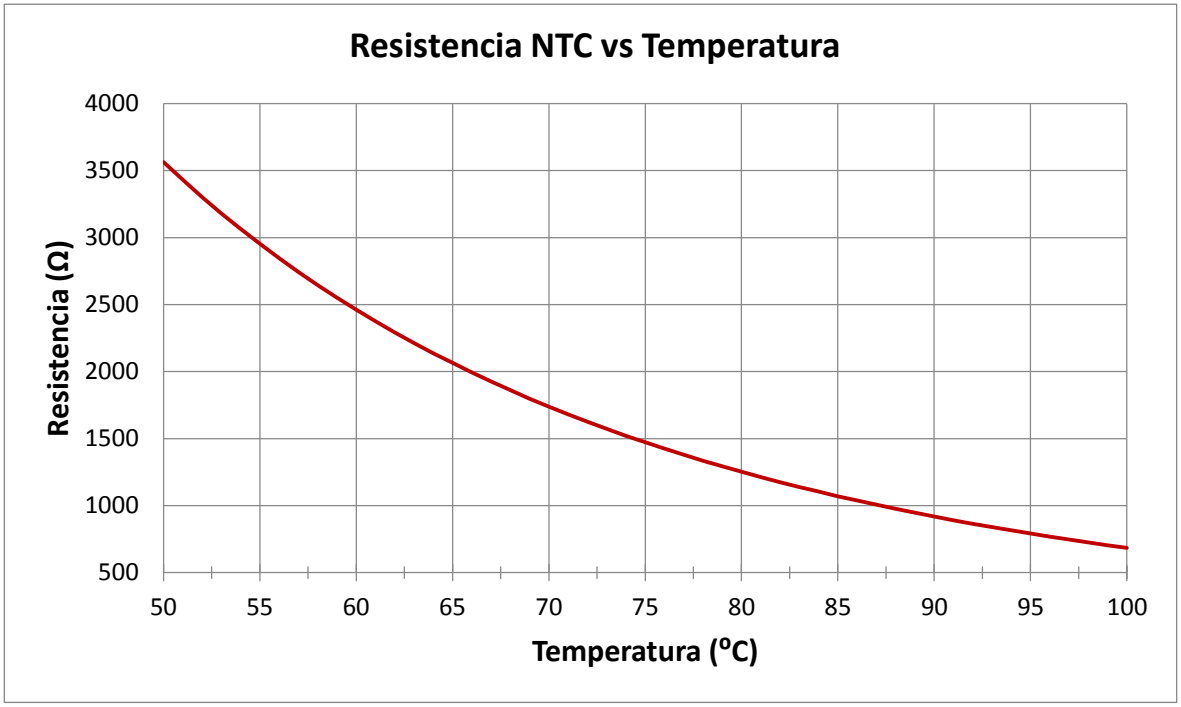


Figura 3.10. Gráfica de la resistencia del termistor en función de la temperatura en un rango concreto.

Para ello se ha intentado maximizar el rango de voltaje a sensar en función de la resistencia secundaria del divisor resistivo (Ecuación 3.2), teniendo en cuenta que la resistencia secundaria debe tener un valor estándar según la serie E24.

$$V_{NTC} = Vcc \frac{R_{SEC}}{R_{SEC} + R_{NTC}} \qquad \textbf{(Eq. 3.2)}$$

Gracias a maximizar el rango del voltaje obtenido tenemos una mayor resolución en la zona deseada de medida. Como se puede observar en la gráfica de la figura 3.12 el valor de 1,6 KΩ nos ofrece un mayor rango de voltaje. Pero esto no nos limita a la hora de utilizar otro valor de resistencia, en la siguiente tabla se comparan diferentes valores de resistencias y su variación numérica en el conversor analógico digital o ADC.

Tabla 3.8. Resistencias secundarias óptimas

Resistencia Secundaria	1800 Ω	**1600 Ω**	**1500 Ω**	1300 Ω
Diferencial de voltaje	1,28283 V	**1,28815 V**	**1,28785 V**	1,27912 V
Diferencial numérico del ADC	1592	**1598**	**1598**	1587

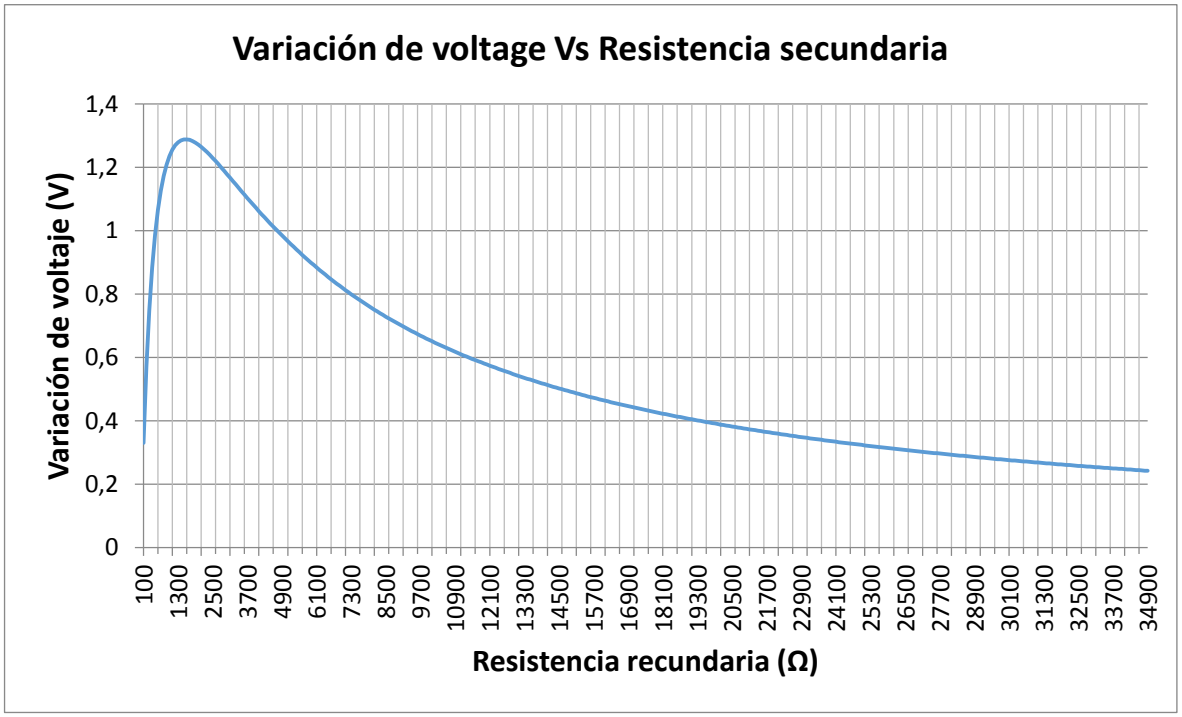


Figura 3.11. Gráfica de la variación de voltaje en función de la resistencia secundaria utilizada.

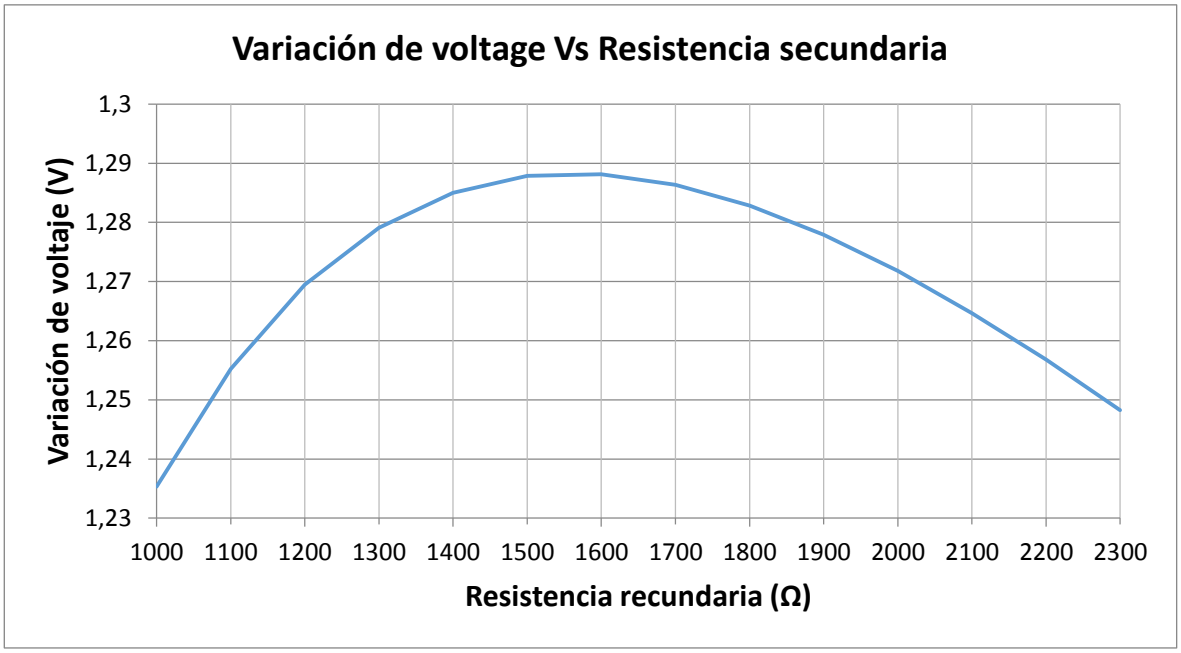


Figura 3.12. Gráfica de la variación de voltaje en función de la resistencia aumentada en la zona de interés.

Finalmente para obtener la temperatura se utiliza la ecuación 3.3 junto con la ecuación 3.2, la cual se ejecutara internamente en el SoC. Obteniendo así el valor de la temperatura interna de funcionamiento.

$$T = \left(\frac{\ln\left(\frac{R_{NTC}}{R_{SEC}}\right)}{\beta} + \frac{1}{T_0} \right)^{-1} \qquad \textbf{(Eq. 3.3)}$$

A modo de comprobación se ha calculado la constante de tiempo del circuito a partir del peor caso, así que mediante Thévenin obtenemos que la resistencia equivalente del circuito obteniendo es el paralelo de las resistencias del divisor de tensión. Finalmente se obtiene:

$$5 \cdot \tau = R \cdot C = 5 \cdot 1\mu F \cdot 1K\Omega = 5ms \qquad \textbf{(Eq. 3.4)}$$

Por la cual cosa el sistema de medición tardara en el peor de los casos 5ms en variar su voltaje, por lo que es un tiempo totalmente aceptable ya que esta medida de temperatura no va a sufrir grandes variaciones en cortos espacios de tiempo.

3.2.5. Zumbador de aviso acústico

El circuito cuenta con un avisador acústico para alertar de la finalización del proceso o de algún fallo en el mismo. Este está formado por un zumbador pasivo de 16 Ohmios, un MOSFET de canal N enriquecido, un diodo en antiparalelo y resistencias que limitan la corriente a través del circuito.

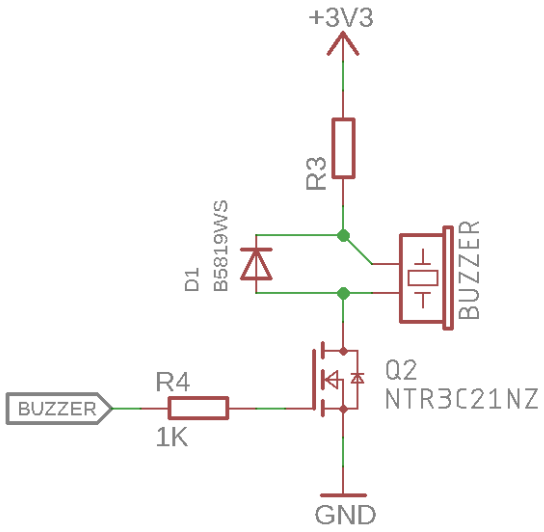


Figura 3.13. Esquemático del zumbador de aviso acústico.

Tabla 3.9. Características principales del MOSFET NTR3C21NZ.

V_{DS}	20 V
$R_{DS(ON)}$ @ 3,3 V	29 mΩ
I_D MAX	3,6 A
Capacitancia de entrada	1540 pF

Tabla 3.10. Características principales del diodo B5819WS.

Tensión de ruptura V_R	40 V
Tensión en directa V_F	0,6 V
Corriente en directa media $I_{F(AV)}$	1 A
Corriente pulsada en directa máxima I_{FSM}	10 A @ 8,3 ms (medio seno)

Para asegurarnos que el MOSFET utilizado trabaja en la zona de saturación se ha calculado lo que tarda su puerta en cargarse a partir de la fórmula 3.5. Consideramos que a partir de 5 veces su constante de tiempo *Tau* su puerta estará cargada y por lo tanto trabajaremos en la zona de saturación.

$$\mathbf{5 \cdot \tau = R \cdot C = 5 \cdot 1450pF \cdot 1K\Omega = 7,7\mu s} \quad \textbf{(Eq. 3.5)}$$

Por lo tanto concluimos que el MOSFET tardara 7,7 microsegundos en encenderse. A partir de este tiempo podemos saber la frecuencia máxima a la cual puede trabajar el MOSFET utilizando la ecuación 3.6.

$$\boldsymbol{Freq. = \frac{1}{5 \cdot \tau} \cong 130\ KHZ} \quad \textbf{(Eq. 3.6)}$$

Finalmente podemos concluir que no tendremos problemas para reproducir un sonido audible con esta configuración debido a que la máxima frecuencia audible por el oído humano está en 20KHz y gracias a nuestra configuración podemos obtener hasta una frecuencia de 130 KHz.

3.2.6. Pantalla de cristal liquido

A modo de extra se ha añadido la posibilidad de poder añadir una pantalla de cristal líquido LCD al conjunto si se desea otra interfaz de control, a parte del uso mediante Wi-Fi. La pantalla LCD es de 2,2 pulgadas de diagonal y utiliza el controlador ili9341. Este tipo de controlador admite varios modos de comunicación pero en este caso se ha decidido utilizar el protocolo de comunicaciones SPI. Hay que tener en cuenta que el modulo LCD comparte el mismo bus de comunicaciones que el chip de lectura del termopar por lo tanto no podrán funcionar a la vez.

Tabla 3.11. Características principales de la pantalla de cristal líquido

Tamaño diagonal	5,58 cm (2,2”)
Resolución	240 x 320 pixeles
Controlador	ILI9341
Profundidad de color	65K o 252K

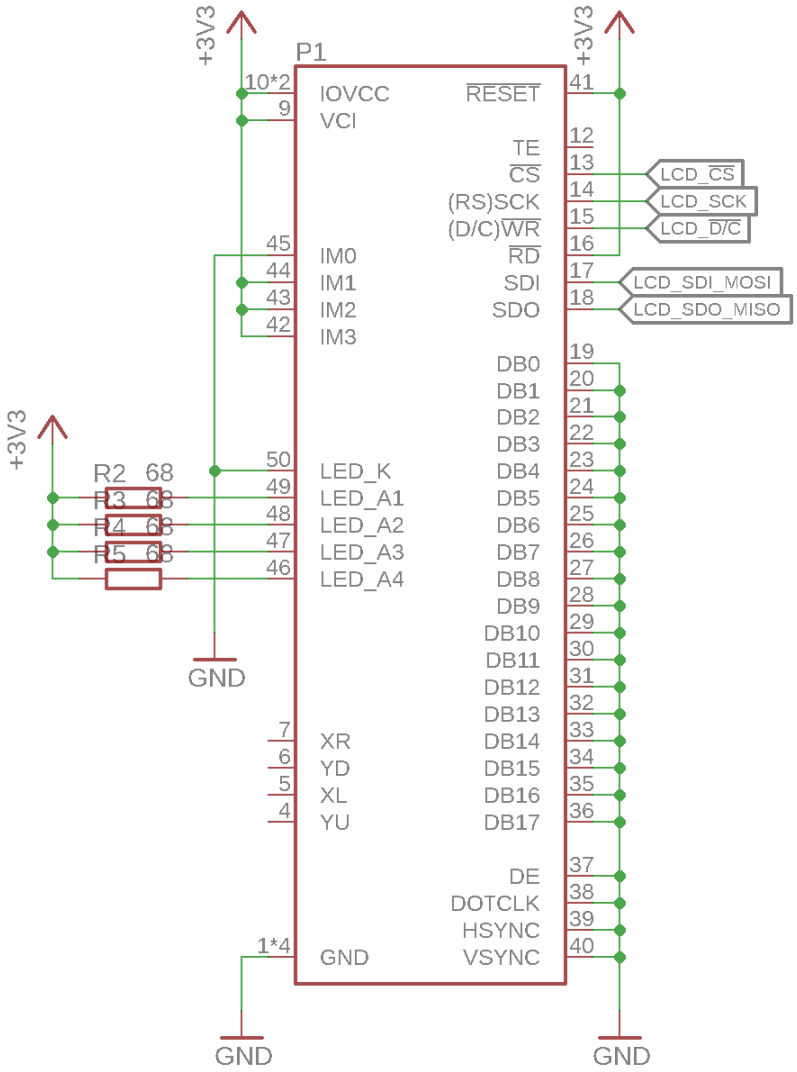


Figura 3.14. Esquemático de la pantalla LCD.

3.2.7. Control táctil capacitivo

Mediante el chip MPR121 del fabricante NXP se ha implementado un sensado de proximidad por capacitancia a través de la carcasa plástica. De esta manera todas las partes metálicas del sistema quedan totalmente aisladas obteniendo un producto más seguro para el usuario final.

Este chip cuenta con 12 entradas capacitivas de las cuales solo se han utilizado 5 y se han añadido dos de reserva para que el usuario final pueda añadir más funciones si este lo cree necesario. Para la comunicación entre el SoC y este chip se ha utilizado el protocolo I_2C de comunicación.

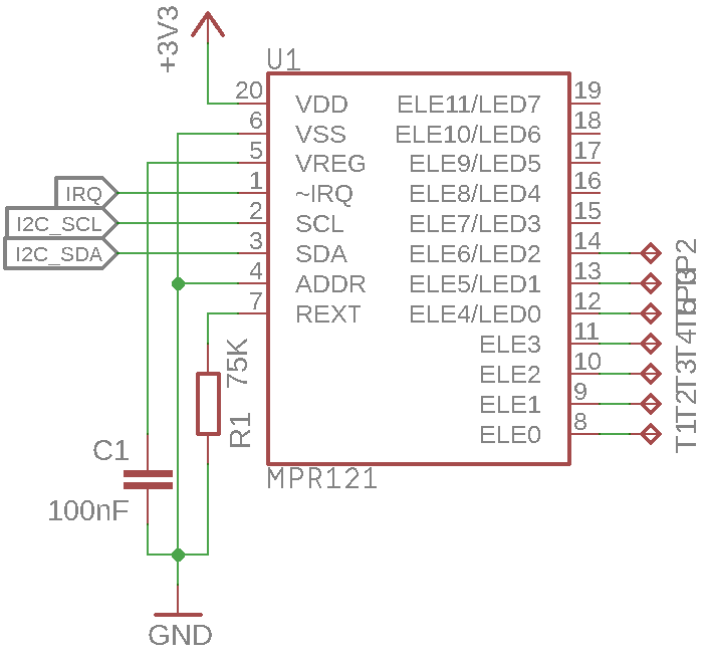


Figura 3.15. Esquemático del controlador táctil capacitivo.

Para la interfaz de entrada se ha diseñado un circuito impreso que integra las conexiones del LCD junto con la interfaz capacitiva. Esta interfaz capacitiva cuenta con 5 electrodos (pads) de medición capacitiva, estos harán la función de botones de entrada para el control del sistema. Más adelante en el apartado de calibración se especifica las diferentes medidas de cada botón y los umbrales de detección de los mismos.

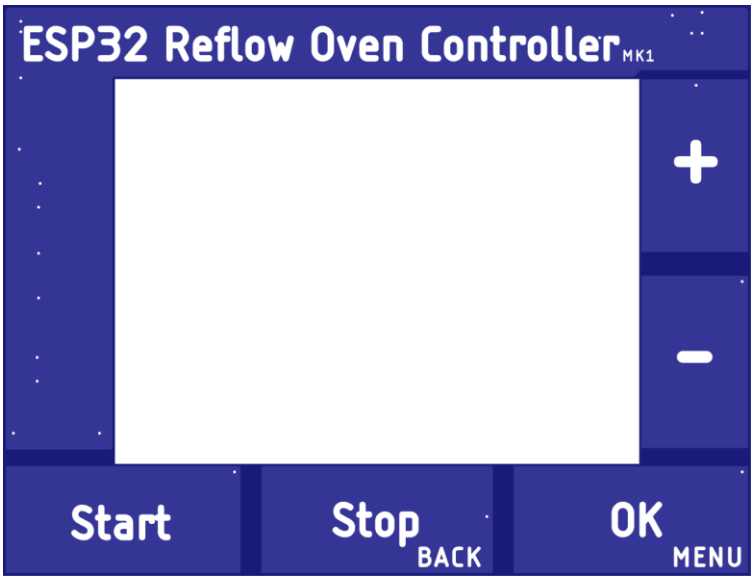


Figura 3.1. Diseño de los electrodos para controlador táctil capacitivo.

3.2.8. Condensadores de desacoplo

Un condensador de desacoplamiento actúa como una reserva de energía. Estos se encuentran lo más cercanos a los circuitos integrados, ya que su trabajo consiste en oponerse a cualquier cambio inesperado en los voltajes de entrada en las líneas de alimentación.

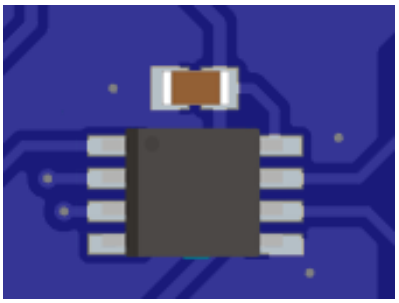


Figura 2.2. Condensador de desacoplo junto al circuito integrado MAX6675.

Estos condensadores trabajan de dos maneras. Si el voltaje de alimentación cae suministrara la energía para intentar mantener el voltaje y si el voltaje se eleva este absorberá la energía, para nueva mente intentar mantener el voltaje constante.

3.3. Selección de otros componentes

En este apartado se especifican los componentes que no pertenecen a la placa principal de control, considerados externos y/o opcionales a la misma.

3.3.1. Conector IEC C13 y C14

Para la alimentación de todo el circuito se ha decidido utilizar un conector del tipo C13 y C14 del estándar IEC (del inglés *International Electrotechnical Commission*) definidos en la especificación IEC 60320.

Cuenta con la entrada de potencia C14 y la salida controlada C13. Esta especificado para soportar 15A y cuenta con un fusible de protección cilíndrico reemplazable.

Figura 3.18. Conector C13 y C14 con fusible integrado.

3.3.2. Interruptor externo

El interruptor de encendido y apagado general, el cual corta toda alimentación del circuito de control y el relé de estado sólido. Este ha sido seleccionado de 16 A para cumplir con la potencia máxima necesaria, el cual cuenta con indicación lumínica para saber el estado del mismo.

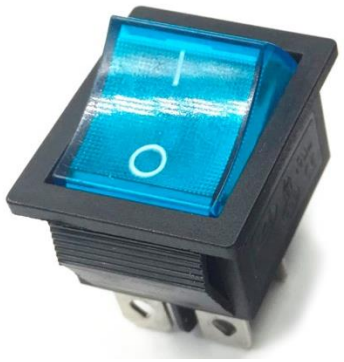


Figura 3.19. Interruptor del tipo “rocker”.

3.3.3. Relé de estado sólido

Un relé de estado sólido, de aquí en adelante SSR (del inglés *solid-state relay*), es la versión electrónica de un relé mecánico. Este no tiene partes móviles por lo que los hace perfecto para la regulación de grandes cargas en frecuencias elevadas, ya que al no tener partes móviles no subirán desgaste por el encendido y apagado constante.

Otro punto a destacar es que su encendido o apagado se le puede considerar instantáneo comparado con un relé mecánico. Pero aun así existen diferentes tipos de SSR, en nuestro caso particular se ha seleccionado un SSR con capacidad de encendido y apagado en el cruce por cero.

Para evitar daños por sobrecargas y según lo especificado se recomiendo utilizar un SSR que sea capaz de soportar corrientes de como mínimo 20A.

Figura 3.20. Rele de estado solido (SSR).

Capítulo 4. Diseño del Software

4.1. Plataforma de desarrollo

La compañía desarrolladora del SoC ESP32, Espressif, ofrece varias herramientas para ser usadas en el desarrollo de sus chips, llamadas SDK (del inglés *Software Development Kit*). Depende del tipo de aplicación que se quiera desarrollar pero todas ellas son de libre uso.

En este proyecto se ha decidido utilizar el "Arduino Core", el cual utiliza el entorno de desarrollo de código libre de Arduino como IDE (del inglés *Integrated Development Environment*) para la programación del SoC.

Los motivos del uso de esta plataforma de desarrollo son la gran comunidad que tiene detrás junto a una mayor documentación disponible comparado con los otros entornos de desarrollo. También es la más utilizada a la hora de desarrollar con estos SoC, de esta manera el comprador final tendrá más facilidades a la hora de modificarse el producto según sus necesidades.

Otro de los puntos fuertes que ofrece la compañía Espressif son las librerías que vienen incluidas con el entorno de desarrollo, gracias a ellas no es necesario la programación a bajo nivel de la aplicación. De esta manera se asegura un desarrollo más fiable, ya que estas librerías las mantiene actualizadas el fabricante y así se evitan fallos causados por una programación incorrecta.

Podemos encontrar el software necesario para su programación en:

- Arduino IDE : https://www.arduino.cc/en/Main/Software
- ESP32 Arduino Core : https://github.com/espressif/arduino-esp32

4.2. Diagrama de flujo de la aplicación

En el diagrama de flujo de la figura 4.1 se ha simplificado el funcionamiento de la aplicación. Se pueden observar los diferentes procesos que se llevan a cabo para el correcto funcionamiento del producto.

Se puede observar cómo se inicializa la aplicación de manera lineal, configurando los diferentes sistemas que integra. Una vez se ha configurado todo el sistema inicial pasa a ejecutar las diferentes tareas en paralelo, detallado más adelante.

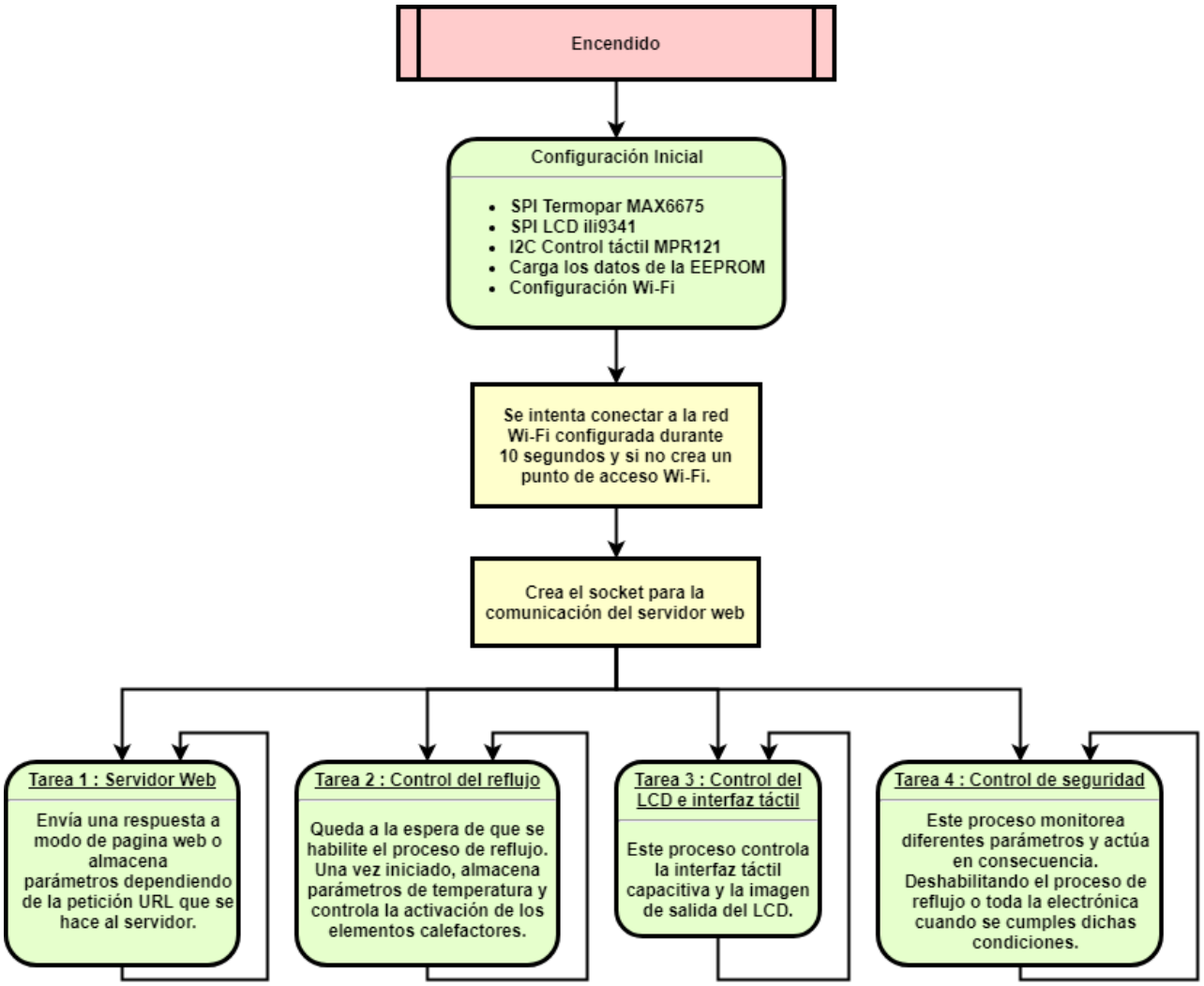


Figura 4.1. Diagrama de flujo de la aplicación.

Teniendo en cuenta que el SoC dispone de dos microprocesadores integrados, de esta manera se dispone de dos núcleos de procesamiento para ejecutar nuestras diferentes tareas.

Para ello se ha decidido utilizar el primer núcleo para ejecutar las tareas 2, 3 y 4. Utilizando el segundo núcleo exclusivamente para las peticiones del servidor web. Ya que este último requiere más carga de trabajo y así se evita que nos ralentice las otras tareas debido a las latencias debidas a las comunicaciones TCP/IP de la comunicación Wi-Fi.

Por otra parte cuando se habla de ejecución en paralelo no es exactamente así. Los dos procesadores son capaces de ejecutar dos tareas en paralelo pero cuando un solo procesador ejecuta varias tareas este va saltando de una a otra creando una falsa sensación de paralelismo.

Para evitar que las tareas 2, 3 y 4 se vean ralentizadas entre ellas se ha evitado realizar bucles de espera. Estos bucles llamados *delay(),* en el lenguaje de programación utilizado, crean un bucle durante un tiempo especificado en el cual el microcontrolador no hace nada más que contar el tiempo para continuar con el código.

Para evitar estas pausas en la ejecución de la aplicación de a decidido utilizar una función de tiempo llamada *millis()* que nos devuelve el tiempo en el que la aplicación lleva ejecutándose en milisegundos. De esta manera se crean intervalos de tiempo en los que se ejecuta una parte u otra en vez de pausar la aplicación.

4.3. Conexión Wi-Fi

La conexión Wi-Fi se hace de dos maneras diferentes. La primera de ellas es conectándose a la red Wi-Fi que se configure mediante el SSID (del inglés *Service Set Identifier*) o comúnmente conocido como nombre de la red Wi-Fi y contraseña de la misma (si la requiere).

Si esta primera conexión falla o no puede conectarse durante los primeros 10 segundos de ejecución, el SoC pasa a crear un punto de acceso Wi-Fi que acepta la conexión de cualquier dispositivo. El punto de acceso cuenta con una opción llamada “portal captivo”, esta genera una notificación automáticamente en los dispositivos indicando que se requiere un inicio de sesión el cual se utiliza para acceder directamente a la página web de control.

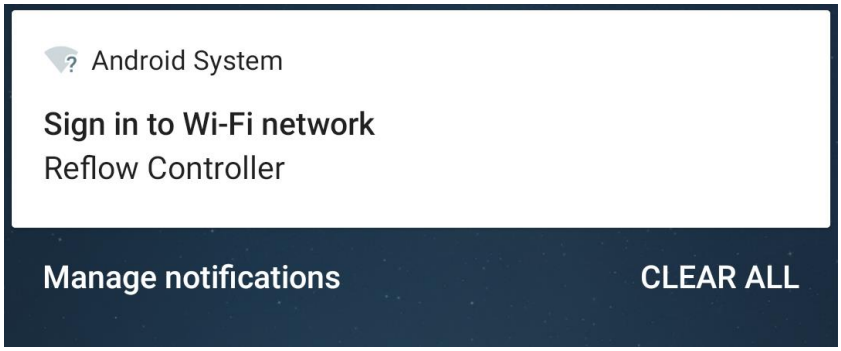


Figura 4.2. Notificación de acceso a la web de control.

4.4. Servidor Web

El servidor web o HTTP (Protocolo de transferencia de hipertexto) es el que se encarga de transferir la información al cliente vía Wi-Fi. Está ha sido la parte más compleja del proyecto, ocupando la mayor parte del código y memoria del microcontrolador, ya que se tenía que incrustar dentro de la programación del microcontrolador los lenguajes de programación de una página web.

Estos están formados por el HTML (del inglés *HyperText Markup Language*) que conforma la estructura del documento, el CSS (del inglés *Cascading Style Sheets*) el cual se encarga del estilo visual del documento y finalmente el JavaScript que ayuda a mejorar la interfaz del usuario.

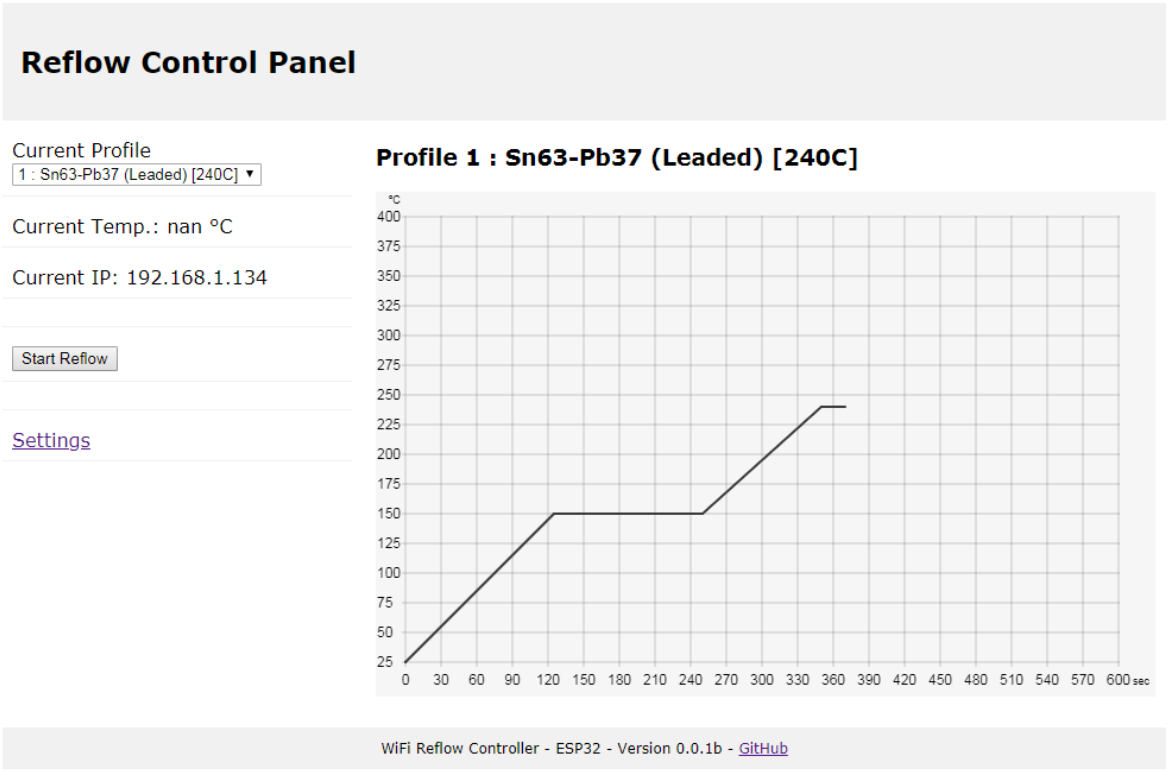


Figura 4.3. Página principal de la web de control.

Para que la web funcione de una manera fluida se ha creado un código lo más simplificado posible pero a la vez que pueda ser intuitivo para el usuario. El diseño utilizado es del tipo adaptativo, que se ajusta a la pantalla en la que se muestra dependiendo del tamaño de la misma.

Este tambien cuenta con un apartado de configuración, donde se puede configurar las temperaturas de los diferentes perfiles de soldadura, la conexión Wi-Fi e reiniciar el sistema.

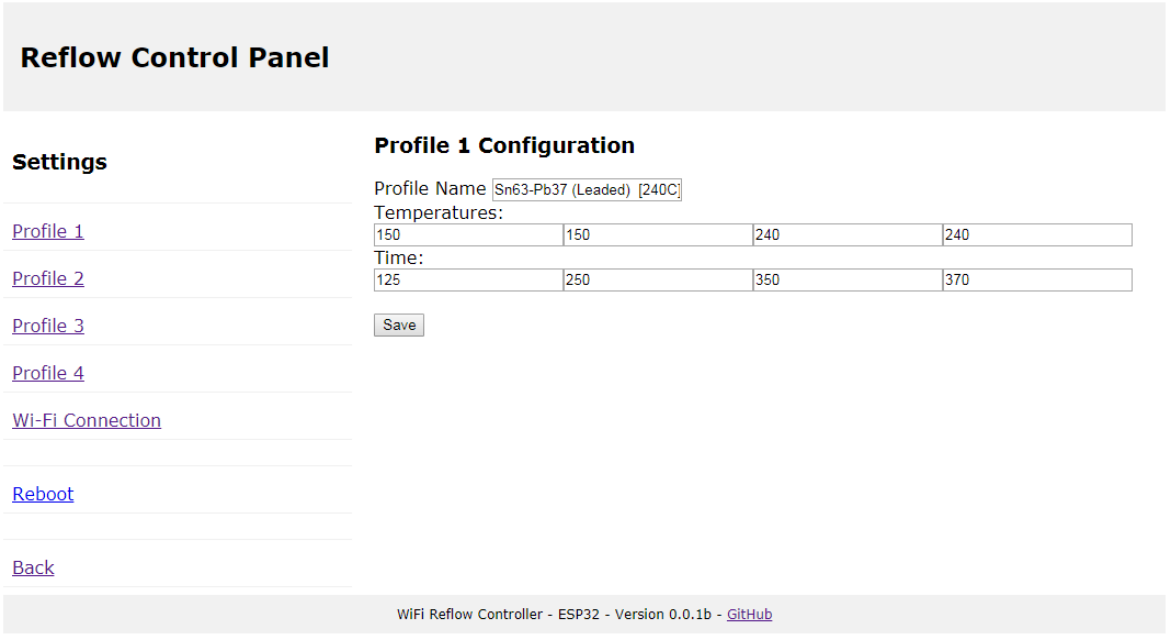


Figura 4.4. Página de configuración de los diferentes parámetros.

4.5. Memoria dinámica no volátil

Para el almacenado de los diferentes parámetros se han reservado 512 Bytes de la memoria flash en los cuales se guardan los nombres de los diferentes perfiles de soldadura, las temperaturas y tiempos de estos. Así como la configuración de la conexión Wi-Fi y el último perfil utilizado.

Esta memoria es cargada en la memoria RAM cada vez que se ejecuta el dispositivo por primera vez y dichos parámetros son actualizados cuando se modifican en el panel de control o se selecciona un nuevo perfil de soldadura.

En la tabla 4.1 se especifica los diferentes parámetros que son almacenados en la memoria no volátil así como su tamaña en bytes y la cantidad de bloques iguales.

Tabla 4.1. Tabla de distribución de la memoria dinámica no volátil

Tamaño	Función
32 Bytes	Wi-Fi SSID
32 Bytes	Wi-Fi Contraseña
32 Bytes x 4 bloques	Nombre de los perfiles 1,2,3 y 4
4 Byte x 16 bloques	Temperaturas de los perfiles
4 Byte x 16 bloques	Tiempos de los perfiles
4 Byte	Perfil de soldadura seleccionado

4.6. Control del proceso de reflujo

Debido a la naturaleza de los elementos calefactores y su lenta respuesta se ha decidido implementado un control PWM (del inglés “Pulse Width Modulation”), para el control del relé de estado sólido, de una frecuencia de 1Hz.

Para el sensado de temperaturas se van tomando una series de 10 muestreos de temperatura para posteriormente poder calcular la variación de temperatura o rampa (en ^{o}C/s).

Por otra parte el proceso de reflujo está constituido por 4 valores de temperatura y tiempo, estos valores se procesan por periodos, obteniendo periodos de rampa y periodos estacionarios. De esta manera se han programado dos tipos de control dependiendo del tipo de periodo.

Para el periodo de rampa de ha implementado un control que regula el ancho del pulso de la señal PWM en función de la diferencia de la rampa a seguir. El pulso PWM se va aumentando o disminuyendo en función de la diferencia entre la rampa a seguir y la rampa actual.

Para los periodos de temperatura estacionarios se ha implementado un control por histéresis, manteniendo siempre la temperatura entre dos valores equidistantes de la temperatura asignada para dicho periodo.

4.7. Control del LCD e interfaz táctil capacitiva

Para la pantalla LCD se ha implementado una interfaz simple. Para ello en la pantalla inicial nos muestra información básica acerca de la conexión que se está estableciendo y la IP de acceso al dispositivo.

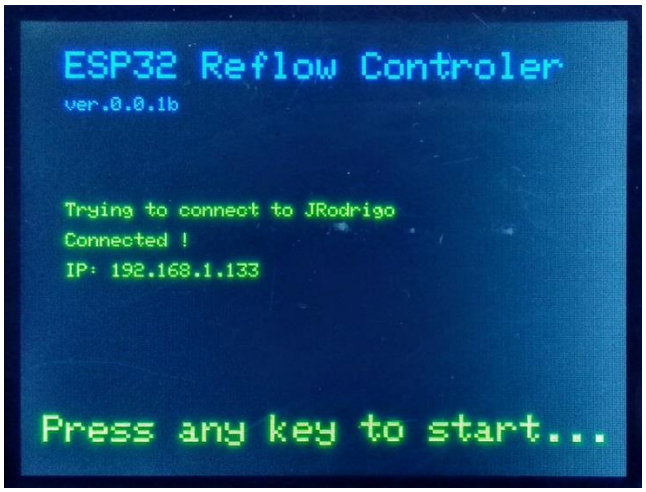


Figura 4.5. Pantalla inicial del controlador.

Para la selección de los perfiles se ha implementado una pantalla en la cual aparece un listado de los perfiles disponibles. Esta lista esta sincronizada con la web de control, por lo que cualquier cambio que efectuemos desde la web afectara al perfil seleccionado en el LCD y viceversa.

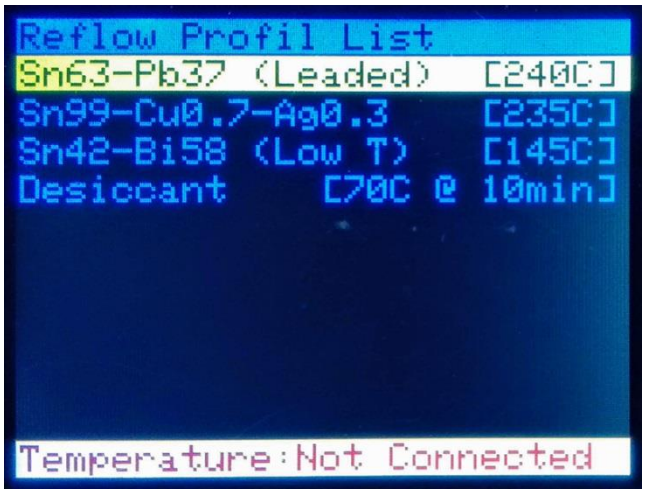


Figura 4.6. Página de configuración de los diferentes parámetros.

Para la navegación a través de la pantalla LCD se ha implementado una interfaz de control táctil capacitiva, con la cual podemos seleccionar entre los diferentes perfiles disponibles o iniciar el proceso de reflujo, en este último caso hace falta mantener "pulsado" durante 2,5 segundos para iniciar el proceso o pararlo.

Para utilizar un mismo código fuente en el caso de utilizar, o no, la pantalla y el control táctil, se ha añadido al inicio una línea de código que permita habilitar o deshabilitar esta expansión. Solo hace falta eliminar o comentar la línea de código que aparece *#define LCD_TOUCH* y de esta manera se deshabilita el uso de la pantalla y el control táctil. De esta manera el compilador no añade dichas funciones y así se puede liberar espacio en la memoria en caso de ser necesario.

4.8. Control de seguridad

Esta subrutina controla diferentes parámetros para asegurarnos que no se sobrepasan valores máximos que pueden llegar a dañar alguna parte del circuito o del mismo horno. Para ello se monitorean la temperatura del circuito de control a través del termistor y la temperatura interna del horno para evitar daños por sobrecalentamiento.

Se han implementado dos tipos de medidas. La primera medida es acústica a través del zumbador integrado en la placa, si la incidencia no es solventada se pasara a deshabilitar el proceso de reflujo hasta que se normalicen las temperaturas.

Capítulo 5. Pruebas y Calibración

5.1. Temperatura de funcionamiento del relé de estado solido

Para asegurarnos del correcto funcionamiento del relé de estado sólido se han realizado unas pruebas de carga para asegurar que no sufre sobrecalentamiento. Teniendo en cuenta que los perfiles de soldadura están diseñados para durar menos de 10 minutos se ha partido de ese tiempo para la realización de diferentes pruebas.

Por otra parte las pruebas están realizadas sin ningún tipo de disipación extra para así simular el peor de los casos posibles.

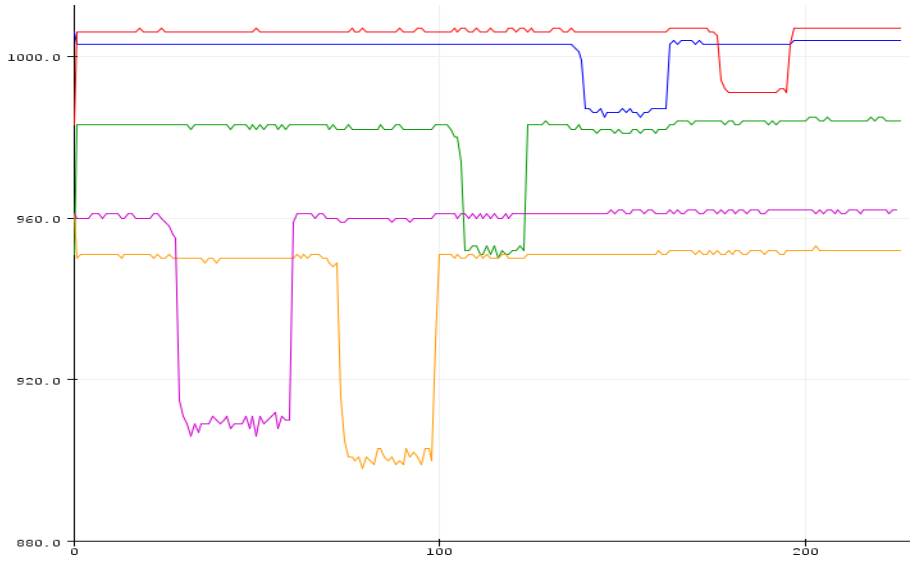
Tabla 5.1. Batería de pruebas realizadas

Tipo de prueba	Tiempo	Temperatura inicial	Temperatura final
Carga constante a 2000 W	10 minutos	19 ^{o}C	49 ^{o}C
Carga constante a 1000 W	30 minutos	49 ^{o}C	55 ^{o}C
Carga constante a 2000 W	10 minutos	55 ^{o}C	70 ^{o}C
Enfriado del SSR	5 minutos	70 ^{o}C	49 ^{o}C
Enfriado del Horno	7 minutos	250 ^{o}C	50 ^{o}C

Teniendo en cuenta que la temperatura de trabajo recomendada por el fabricante es de como máximo 80 ^{o}C podemos concluir que no se ha superado en ninguno de los casos. Otro factor a tener en cuenta es que el SSR no trabaja de forma constante como en las pruebas realizadas, si no que efectúa una regulación activándose y desactivándose, por la cual cosa mientras esta desactivado se puede enfriar y así obtener unas temperaturas de funcionamiento muy inferiores a las aquí obtenidas.

5.2. Calibración del control táctil capacitivo

Para la calibración de los umbrales de detección del teclado capacitivo lo primero que se efectuó es una medida en bruto de los diferentes valores de cada electrodo de medida, para así poder discriminar entre el ruido y los valores de detección.



Leyenda de los electrodos: Start, Stop, OK, menos "-" y más "+".

Figura 5.1. Lectura de los diferentes botones capacitivos.

Para la correcta calibración hay que entender el tipo de control capacitivo que efectúa el chip MPR121 para la detección. Este detecta las pulsaciones a partir de dos umbrales, un primer umbral de pulsación (*Touch Threshold*) y un segundo umbral de liberación (*Touch Release*).

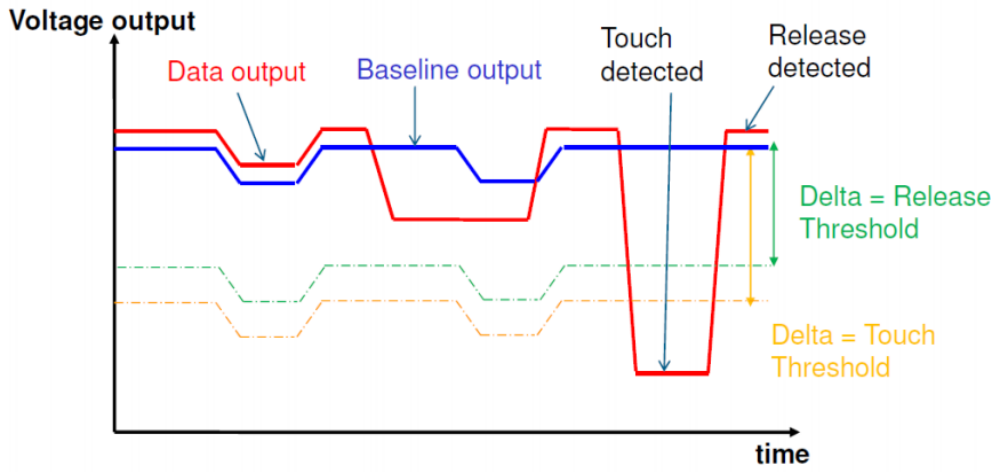


Figura 5.2. Detección de la pulsación del chip MPR121 (Fuente: Datasheet)

Tras varias pruebas finalmente se decidió utilizar un valor de 12 para el umbral de pulsación y 6 para el umbral de liberación, ya que son los valores que mejor detectan las diferentes pulsaciones.

Por otra parte el controlador táctil cuenta también con la posibilidad de poder configurar los umbrales de detección independientemente para cada electrodo. De esta manera si alguno de los electrodos no consiguiera detectar correctamente las pulsaciones se podría ajustar de manera individual para no afectar al resto.

5.3. Calibración del termopar y el termistor NTC

La calibración de las temperaturas se ha efectuado mediante una referencia de temperatura conocida, en este caso se ha utilizado un multímetro con función de medida de temperatura. Para ello se han colocado la sonda de temperatura del multímetro y el sensor a calibrar en un recipiente cerrado.

Una vez los valores de temperatura son estables y sin variación, se ha modificado el software con la variación pertinente y se ha vuelto a efectuar la medición. Finalmente cuando los valores medidos por el multímetro y el sensor son iguales se ha dado por buena la calibración.

5.4. Calibración del control de reflujo

Para la calibración del proceso de reflujo se tiene que obtener de manera empírica el *overshoot* (sobrerreacción) del sistema, esta es debida a la inercia térmica del propio sistema. Para ello se ha hecho funcionar el controlador sin ningún tipo de valor que corrija el *overshoot,* figura 5.3., en la cual se puede observar como la temperatura (rojo) no se ajusta al perfil de soldadura.

Tras probar analíticamente varios valores de temperatura para mitigar el *overshoot*, se ha obtenido que el valor óptimo es 15 ^{o}C. Como se puede observar en la figura 5.4 si añadimos una corrección de 15 ^{o}C se mitiga el problema ocasionado por el *overshoot*.

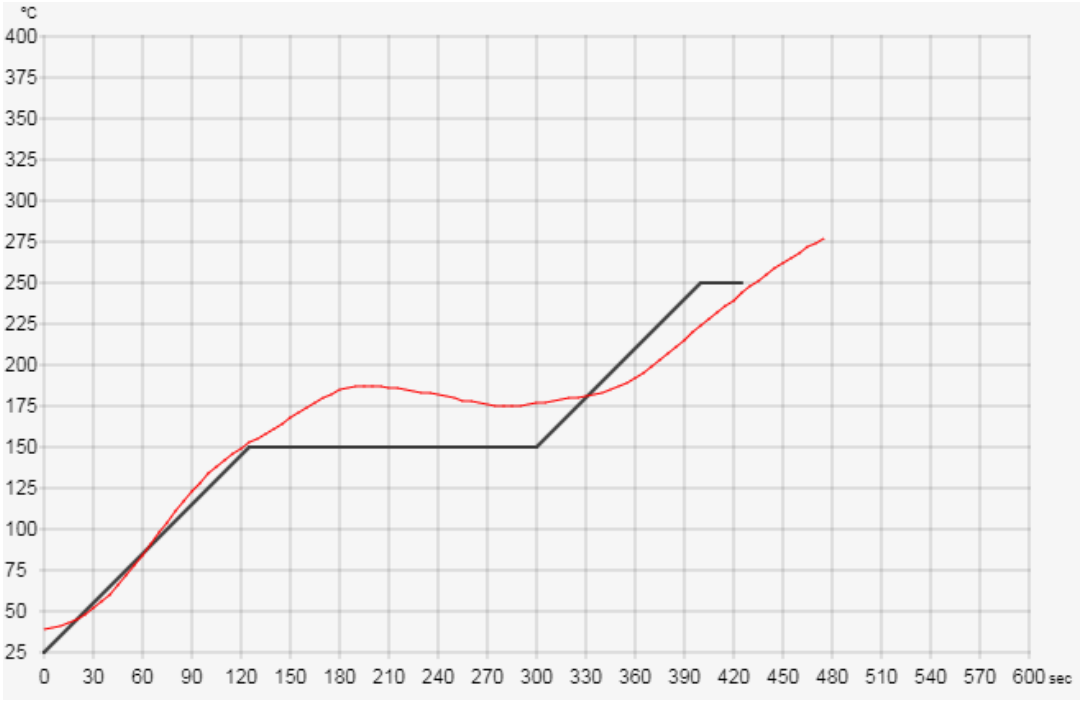


Figura 5.3. Proceso de reflujo sin control del *overshoot*

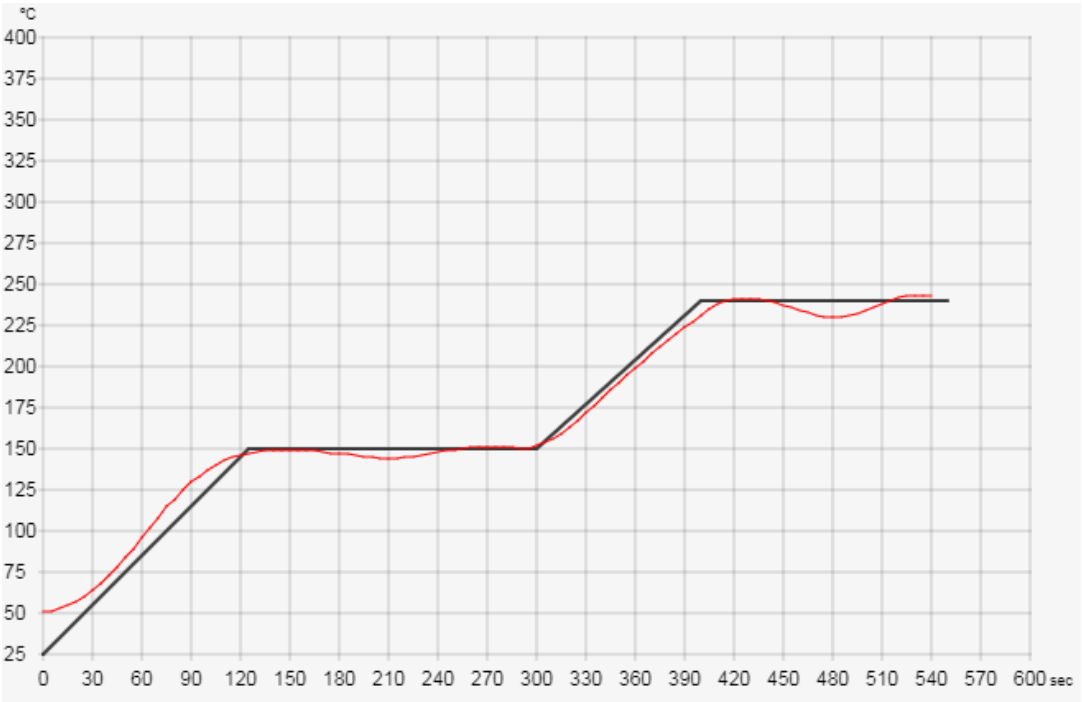


Figura 5.4. Proceso de reflujo con control del *overshoot*

Capítulo 6. Análisis del impacto ambiental

Para la selección de todos los componentes de este proyecto se ha tenido en cuenta la normativa RoHS (de las siglas en inglés *Restriction of Hazardous Substances*) referida a la directiva 2002/95/CE que restringe el uso de diez sustancias peligrosas en aparatos eléctricos y electrónicos.

- Plomo (Pb): < 1000 ppm (0.1%)
- Cadmio (Cd): < 100 ppm (0.01%)
- Mercurio (Hg): < 1000 ppm (0.1%)
- Cromo hexavalente (Cr VI) < 1000 ppm (0.1%)
- Bifenilo polibromado (PBB): < 1000 ppm (0.1%)
- Éteres de difenilo polibromados (PBDE): < 1000 ppm (0.1%)
- Bis(2- Etilhexilo) ftalato (DEHP): < 1000 ppm (0.1%)
- Bencilo butil stalato (BBP): < 1000 ppm (0.1%)
- Dibutil ftalato (DBP): < 1000 ppm (0.1%)
- Ftalato de disobutilo (DIBP): < 1000 ppm (0.1%)

El producto final deberá ir identificado con el símbolo de la figura 5.1 para informar a los consumidores que este tipo de producto no se puede tirar a la basura, sino que debe ser depositado en un punto selectivo de recogida de residuos electrónicos.

Figura 6.1. Símbolo de recogida selectiva.

Conclusiones

Se puede concluir que se han cumplido con los objetivos marcados del proyecto, consiguiendo implementar un controlador para la soldadura mediante reflujo en hornos comunes de sobremesa y de esta manera hacer este tipo de procesos de soldadura más accesibles al reducir su coste.

A título personal me ha servido para aumentar mis conocimientos sobre este tipo de controladores, pero sobre todo me ayudado a ampliar mis conocimientos en programación, debido a la gran variedad de lenguajes diferentes que he necesitado utilizar para la implementación de este proyecto.

A modo de crítica constructiva cabe decir que hay que seguir trabajando en la programación para encontrar más fallos y corregirlos. También hay que trabajar en la página web y la pantalla LCD, para crear una interfaz más rica e intuitiva visualmente.

Finalmente concluir con el tipo de control utilizado. Se planteó durante el proyecto el uso de un control PI pero no se llegó a poder implementar por falta de tiempo y por otra parte los parámetros obtenidos solo serían precisos para el horno modelizado. Para resolver esto se tendría que implementar un proceso de auto-configuración de manera que el controlador ejecute una batería de pruebas y así obtener los valores de configuración dependiendo del horno conectado.

Presupuesto y Análisis Económico

En este capítulo se especifican los costes (sin impuestos) asociados al desarrollo de este proyecto, los cuales se han repartido en cuatro puntos: los costes de ingeniería, los costes indirectos, los costes de los materiales y los costes de producción.

Costes de ingeniería

Los costes de ingeniería provienen de la inversión en el personal para el desarrollo del proyecto. En el cual se especificas las horas invertidas en las diferentes tareas del mismo. Para el coste de las horas se ha tenido en cuenta el convenio de la industria siderometalúrgica de la provincia de Barcelona para el año 2019 para un ingeniero técnico recién licenciado.

Tabla P.1. Presupuesto de ingeniería

Trabajo	Coste / Hora	Horas	Total
Estudio previo	12 €	60	720 €
Diseño del hardware	12 €	160	1920 €
Diseño del software	12 €	260	3120 €
Montaje del prototipo	12 €	16	192 €
Pruebas y calibración	12 €	80	960 €
Memoria técnica	12 €	100	1200 €
	TOTAL	676	8112 €

Costes indirectos

Los costes indirectos derivan de diferentes partes, tales como el uso de un ordenador o el equipamiento de laboratorio, licencias de software o material adicional a los especificados en el apartado de materiales.

Tabla P.1. Presupuesto de costes indirectos

Descripción	Total
Ordenador de sobremesa	264 €
Equipamiento de laboratorio	132 €
Materiales secundarios	50 €
Placas de desarrollo para el estudio previo	58 €
Licencia de software Eagle	133 €
Piezas impresas en 3D	20 €
TOTAL	657 €

Costes de los materiales

En este apartado se especifican los componentes necesarios para producir las diferentes partes que componen este proyecto. Cada tabla equivale a una unidad final que se han repartido en los diferentes productos que se quiere ofrecer.

Tabla P.3. Listado de los materiales y costes de la placa de control

Part No.	Descripción	Cantidad	Coste Unitario	Total
ESP32-WROOM-32	Modulo microcontrolador con Wi-Fi incluido	1	2,54 €	2,54 €
HLK-PM03	Fuente de alimentación AC 230 V a DC 3,3 V	1	2,03 €	2,03 €
MAX6675	Lector de termopar tipo K mediante SPI	1	8,71 €	8,71 €
R0805-100	Resistencia SMD de 100 Ω con encapsulado 0805	1	0,01 €	0,01 €
R0805-1K	Resistencia SMD de 1 KΩ con encapsulado 0805	2	0,01 €	0,02 €
R0805-10K	Resistencia SMD de 10 KΩ con encapsulado 0805	3	0,01 €	0,03 €
R0805-1K5	Resistencia SMD de 1,5 KΩ con encapsulado 0805	1	0,01 €	0,01 €
C0805-100nF	Condensador SMD de 100 nF con encapsulado 0805	2	0,02 €	0,04 €
C0805-1uF	Condensador SMD de 1 µF con encapsulado 0805	3	0,05 €	0,15 €
NTCLE100E3	Termistor NTC 10Kohms radial	1	0,53 €	0,53 €
UMH1N	Dos transistores bipolares prepolarizados	1	0,32 €	0,32 €
10D391K	Varistor de protección de entrada	1	0,42 €	0,42 €
FUSE	Portafusibles con fusil de cristal cilíndrico de 200 mA	1	1,80 €	1,80 €
BUZZER	Zumbador pasivo de 16 Ω	1	0,53 €	0,53 €
SCREWBLOCK	Clema de 2 pines de conexión de 5 mm	2	0,32 €	0,64 €
PINHEAD	Conector para cables de 2,54 mm	1	1,16 €	1,16 €
TPCON	Conector para el termopar impreso en 3D	1	2,00 €	2,00 €
PCB	Placa de circuito impreso	1	1,15 €	1,15 €
			TOTAL	22,09€

Tabla P.4. Listado de los materiales y costes de la placa del LCD más control táctil capacitivo

Part No.	Descripción	Cantidad	Coste Unitario	Total
LCD	2.2" 240x320 Pantalla LCD driver ILI9341	1	8,33 €	8,33 €
MPR121	Driver controlador táctil por I2C	1	1,16 €	1,16 €
R0603-50	Resistencia SMD de 50 Ω con encapsulado 0603	4	0,01 €	0,04 €
R0805-75K	Resistencia SMD de 75 KΩ con encapsulado 0805	1	0,01 €	0,01 €
C0805-100nF	Condensador SMD de 100nF con encapsulado 0805	3	0,02 €	0,06 €
C0805-1uF	Condensador SMD de 1 µF con encapsulado 0805	1	0,05 €	0,05 €
PINHEAD	Conector para cables de 2,54 mm	1	1,16 €	1,16 €
PCD	Placa de circuito impreso	1	0,85 €	0,85 €
			TOTAL	11,66€

Tabla P.5. Listado de los materiales y costes para el montaje en caja

Part No.	Descripción	Cantidad	Coste Unitario	Total
BOX	Caja ABS IP65 158 x 90 x 60 mm	1	10,82 €	10,82 €
KCD4-202	Interruptor con piloto de encendido tipo "rocker" AC 250V 15A	1	1,25 €	1,25 €
SOCKET	Entrada y salida eléctrica tipo C13 C14 según IEC320	1	3,55 €	3,55 €
SSR-50A	Relé de estado sólido 50A, activación DC, cruce por cero	1	13,75 €	13,75 €
TF48	Terminal de crimpado faston aislado 4,8 mm	5	0,16 €	0,80 €
TF63	Terminal de crimpado faston aislado 6,3 mm	4	0,16 €	0,64 €
TC53	Terminal de crimpado circular ID 5,3mm	4	0,11 €	0,44 €
HEATSINK	Disipador de aluminio a medida	1	7,00 €	7,00 €
			TOTAL	38,25 €

Costes de producción

Este apartado se especifica los costes derivados por la producción del producto. Estos gastos se producen por cada unidad ensamblada y se ha desglosado en las diferentes partes del mismo. Para ello se ha tenido en cuenta la producción propia con los recursos actuales de los que dispongo.

Tabla P.6. Presupuesto de ingeniería

Trabajo	Coste / Hora	Horas	Total
Ensamblado y testeo placa de control	12 €	0,5	6,00 €
Ensamblado y testeo placa del LCD	12 €	0,2	2,40 €
Ensamblado y testeo del producto en caja	12 €	1	12,00 €
	TOTAL	1,7	20,40 €

Análisis Económico

En este último apartado se analizara la viabilidad económica del producto en función de los costes mencionados anteriormente. Teniendo en cuenta que los costes de ingeniería e indirectos, que estos ascienden a 8769 €, solo se producen una vez y deberán serán sufragados por las ventas del producto.

Como se mención con anterioridad el objetivo es ofrecer diferentes productos para optar a un mercado de usuarios más amplio.

Para ello se ha calculado los precios finales de los artículos (sin IVA) con una proporción de 2,5 frente a los costes de los componentes. El benefició por unidad serían las ganancias por unidad eliminando los gastos relacionados con los componentes y producción.

Tabla P.7. Precios de venta final y ganancias

Producto	Precio de venta	Beneficio	Retorno en
Placa de control	59,95 €	31,86 €	275 unidades
Placa del interfaz LCD y táctil	29,95 €	15,89 €	552 unidades
Producto totalmente ensamblado	185,95 €	92,55 €	95 unidades

La columna de retorno de la tabla anterior nos muestra la cantidad de unidades que tenemos que vender para poder sufragar los gastos de ingeniería e indirectos generados por el desarrollo del producto. Una vez que se hayan vendido esas unidades se producirá un beneficio el cual se puede reinvertir en ofrecer por ejemplo actualizaciones del software.

Finalmente para un plan de lanzamiento inicial del producto hay que estimar una cantidad concreta de unidades a producir, para así reducir ligeramente los costes al realizar pedidos en grandes cantidades. En la siguiente tabla se hace una recomendación de las unidades necesarias para una primera producción que cubren los gastos de ingeniería e indirectos.

Tabla P.8. Precios de venta final y ganancias

Producto	Retorno
Placa de control	175 unidades
Placa del interfaz LCD y táctil	40 unidades
Producto totalmente ensamblado (incluye placa de control y LCD)	30 unidades

Bibliografía

Biografía consultada

Horowitz, P., Winfield , H. (2015) "The Art of Electronics" New Jork: Cambridge University Press

Mohan, N., Undeland, T. M., Robbins, W. P. (2002) "Power Electronics: Converters, Applications, and Design" Nueva Jersey: John Wiley & Sons Inc

Franco, S. (2014) "Design With Operational Amplifiers And Analog Integrated Circuits" San Francisco: McGraw-Hill

Flanagan, D. (2011) "JavaScript: The Definitive Guide, 6th Edition" California: O'Reilly Media

Espressif (2018) "ESP32 - Technical Reference Manual" revisión 4.0 disponible en: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

Espressif (2019) "ESP32 - Hardware Design Guidelines" revisión 2.7 disponible en: https://www.espressif.com/sites/default/files/documentation/esp32_hardware_design_guidelines_en.pdf

Nota de aplicación AN10365 del fabricante NPX (2012) "Surface mount reflow soldering" revisión 6 disponible en: https://www.nxp.com/docs/en/application-note/AN10365.pdf

Normativas

- RoHS, Directiva 2002/95/CE de Restricción de ciertas Sustancias Peligrosas en aparatos eléctricos y electrónicos, adoptada en febrero de 2003 por la Unión Europea. https://www.boe.es/doue/2003/037/L00019-00023.pdf
- CEM, Directiva 2004/108/CE sobre compatibilidad electromagnética, que garantiza que los dispositivos funcionen satisfactoriamente en presencia de otras fuentes electromagnéticas y a la vez que no afecten a otros sistemas del entorno que los rodea. http://www.f2i2.net/documentos/lsi/Dir_2004-108.pdf
- REBT, Real Decreto 842/2002 es el Reglamento Electrotécnico para Baja Tensión de España que tiene como objeto establecer el marco de las condiciones técnicas y garantías que deben reunir las instalaciones eléctricas conectadas a una fuente de suministro en los límites de baja tensión. En este se incluyen las Instrucciones Técnicas Complementarias (ITC-BT). http://www.f2i2.net/documentos/lsi/rbt/guias/guia_bt_rd_842_02_sep03r1.pdf

Hojas de datos (Datasheets)

- ESP32 Series (ESP32-D0WD, ESP32-D0WDQ6, ESP32-D2WD, ESP32-S0WD) https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
- HLK-PM03 Ultra-Compact Module Power https://www.mikrocontroller.net/attachment/349613/HLK-PM03.pdf
- 10D391K – Metal Oxide Varistor https://www.bourns.com/data/global/pdfs/MOV10D.pdf
- MAX6675 – Cold-Junction-Compensated K-Thermocoupleto-Digital Converter https://datasheets.maximintegrated.com/en/ds/MAX6675.pdf
- NTCLE100E3 – NTC Thermistor https://www.vishay.com/docs/29049/ntcle100.pdf
- ILI9341 – TFT Single Chip Driver 240x320 https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf
- MPR232 – Proximity Capacitive Touch Sensor Controller https://www.nxp.com/docs/en/data-sheet/MPR121.pdf

Anexo A – Esquemáticos y Hardware

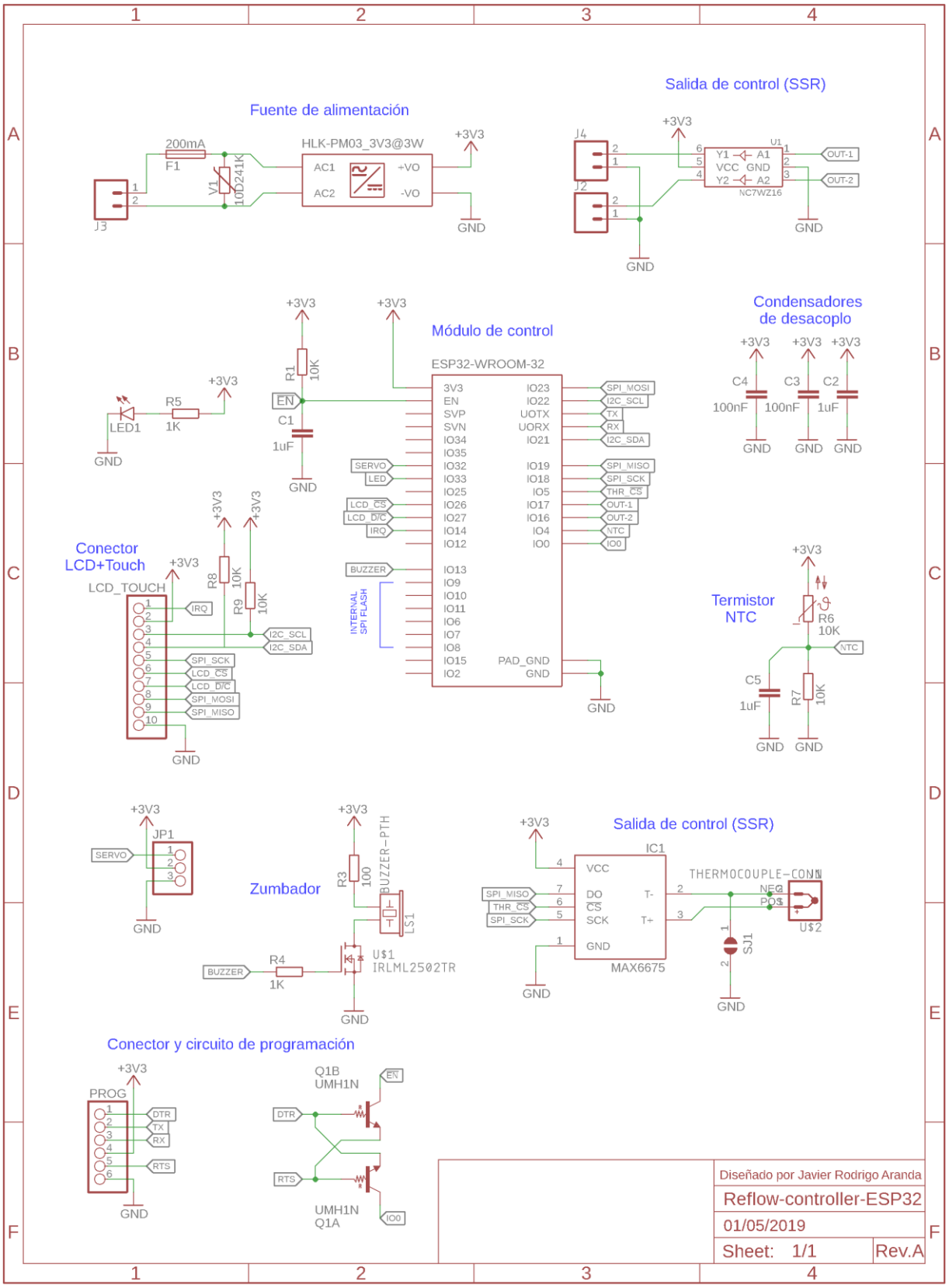
En este anexo A podréis encontrar los esquemas de diseño del controlador de reflujo y el modulo LCD táctil. También se han añadido los renderizados de los circuitos impresos, tanto de la parte superior como inferior del circuito impreso.

Para el desarrollo de la parte del software se decidió utilizar un repositorio online, para de esta manera se tienen controlados todos los cambios efectuados en el código fuente facilitando el trabajo de programación. Finalmente en este repositorio también están disponibles los archivos de diseño, tanto el esquemático como el circuito impreso para ser usados en *Autodesk Eagle*.

Enlace al repositorio de archivos:

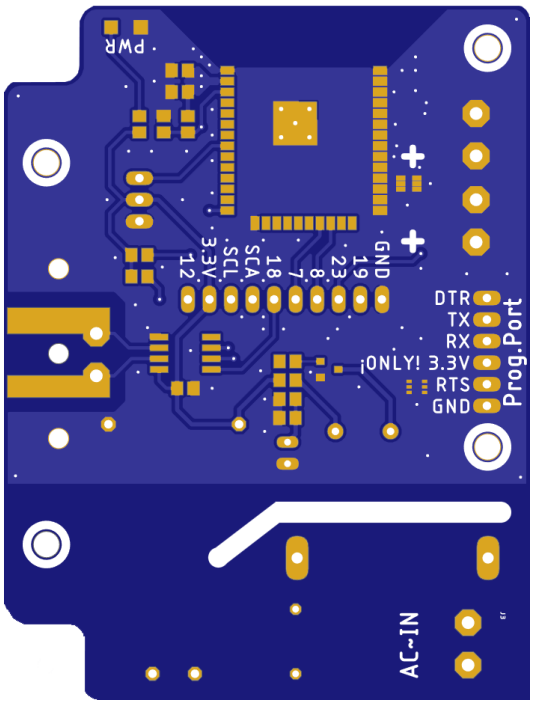
https://github.com/JRodrigoTech/ESP32-Reflow-Toaster-Oven-Controller

A1. Esquema de la placa principal de control

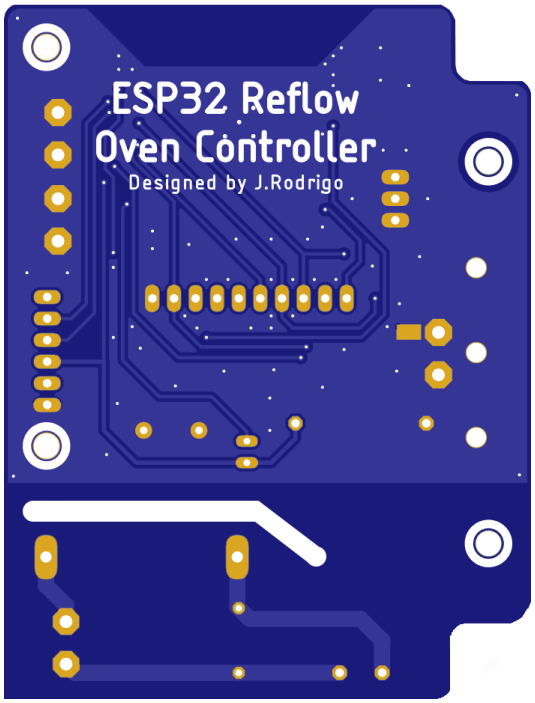


A2. Placa principal de control

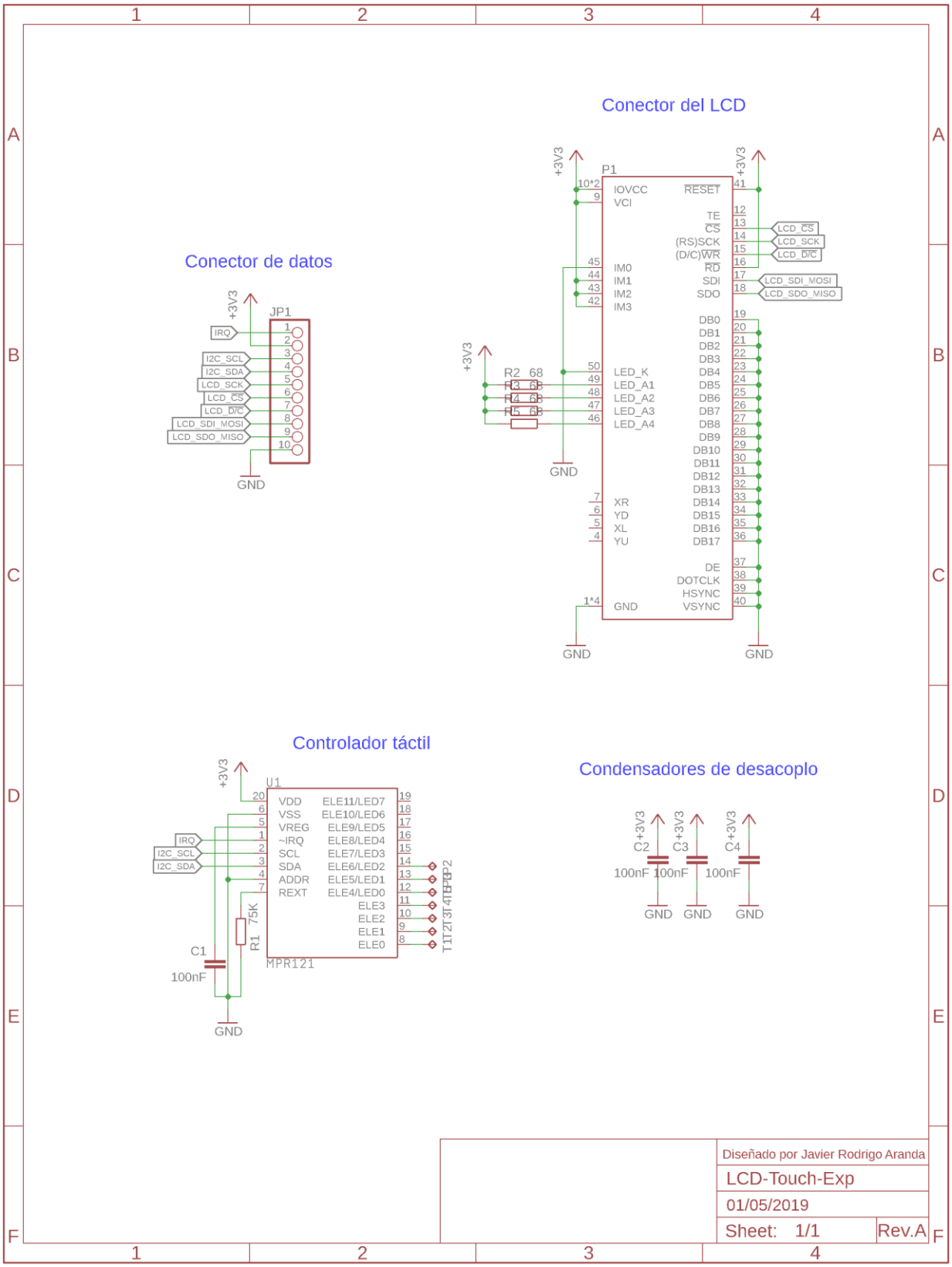
Vista superior



Vista inferior



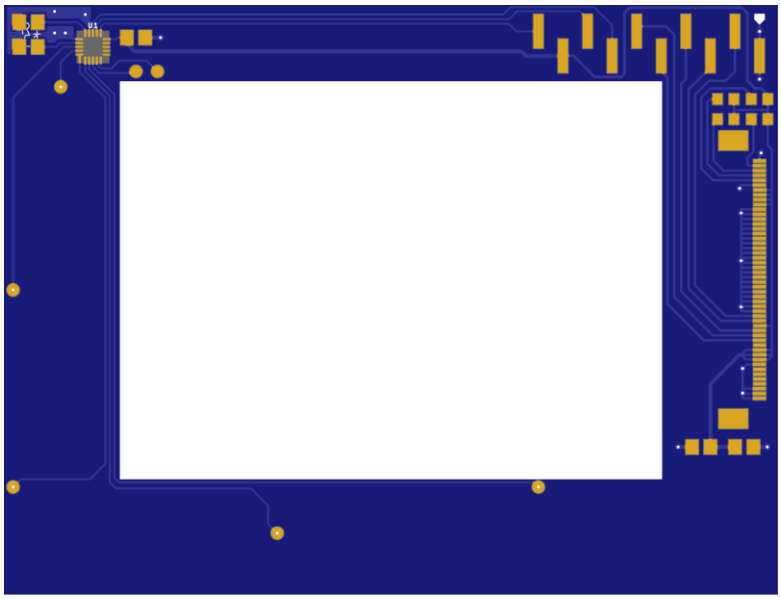
A3. Esquema de la placa tàctil capacitiva y el LCD



A4. Placa tàctil capacitiva y el LCD

Vista superior

Vista inferior



Anexo B – Programación

B1. MAIN.ino

```
001   #include <WiFi.h>
002   #include <ESPmDNS.h>
003   #include <DNSServer.h>
004   #include <WebServer.h>
005   #include <EEPROM.h>
006   #include <esp_task_wdt.h>
007   #include "max6675.h"
008   #include "Variables.h"
009   #include "EEPROM.h"
010   #include "html.h"
011   #include "SPI.h"
012
013   // Enable LCD + Touch Control (Comment to disable)
014   #define LCD_TOUCH
015
016   // Serial Debug
017   #define SERIAL_DEBUG
018
019   // Initialize Termocouple
020   MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);
021
022   // Thermocouple String back function
023   String thermocouple_temp() {
024     return String(thermocouple.readCelsius());
025   }
026
027   #ifdef LCD_TOUCH
028     // LCD SCREEN
029     #include "Adafruit_GFX.h"
030     #include "Adafruit_ILI9341.h"
031     Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_MOSI,
    TFT_CLK, TFT_RST, TFT_MISO);
032     #include "LCD.h" // LCD Print Functions
033     // MPR121 I2C Touch Controller
034     #include <Wire.h>
035     #include "Adafruit_MPR121.h"
036     Adafruit_MPR121 touch = Adafruit_MPR121();
037   #endif
038
039   void setup() {
040     #ifdef SERIAL_DEBUG
041     Serial.begin(115200);    // SERIAL DEBUG
042     #endif
043     // Pre-load EEPROM DATA
044     loadEEPROMdata();
045
046   #ifdef LCD_TOUCH
047     // Start the LCD
```

```
048     tft.begin();
049     tft.setRotation(3);
050     tft.fillScreen(ILI9341_BLACK);
051     // Print welcome text during the start up
052     tft.setTextColor(ILI9341_BLUE);
053     tft.setTextSize(2);
054     tft.println("");
055     tft.println("  ESP32 Reflow Controler");
056     tft.setTextSize(1);
057     tft.println("");
058     tft.println("    ver.0.0.1b");
059     tft.setTextSize(3);
060     tft.println("");
061     tft.println("");
062     tft.setTextSize(1);
063     tft.setTextColor(ILI9341_GREEN);
064     tft.print("    Trying to connect to ");
065     tft.println(ssid);
066
067
068     // Initialize Touch Controller
069     touch.begin(0x5A);
070
071   #endif
072
073     // SSR POWER OUTPUT
074     pinMode(SSR_PIN, OUTPUT);
075     digitalWrite(SSR_PIN, LOW);
076
077     // STATUS LED
078     pinMode(STATUS_LED, OUTPUT);
079     digitalWrite(STATUS_LED, HIGH);
080
081     // Buzzer control
082     ledcSetup(0, 2000, 8);    // PWM Ch, Freq, Resolution
083     ledcAttachPin(13, 0);     // Pin , PWM Ch
084
085     if(current_profile > 4 || current_profile < 1 ){
    current_profile=1; }   // Profile Range Delimiter
086
087     // WIFI CLIENT MODE //
088     WiFi.begin(ssid, password);
089
090     while (WiFi.status() != WL_CONNECTED && AP_MODE == false) {
091         delay(500);
092         digitalWrite(STATUS_LED, !digitalRead(STATUS_LED));
093         time_out = time_out + 500 ;
094         if(time_out >= conn_time_out){
095           AP_MODE = true;
096         }
097     }
098
099     // WIFI AP MODE if not CLIENT //
100     if(AP_MODE == false){    // as CLIENT
101
102       MDNS.begin("reflow"); // Bonjour Service http://reflow.local/
103
104     }else if(AP_MODE == true){
```

```
105
106       // AP WIFI MODE //
107       WiFi.mode(WIFI_OFF); // Close the WiFi client connection
108       delay(1000);
109
110       // Configure the Access Point
111       WiFi.mode(WIFI_AP);
112       WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
113       WiFi.softAP("Reflow Controller"); // Without password
114       //WiFi.softAP("Reflow Controller", "1234");   //  With password
115
116       // if DNSServer is started with "*" for domain name, it will
117       // reply with provided IP to all DNS request
118       dnsServer.start(DNS_PORT, "*", apIP);
119
120     }
121
122   #ifdef LCD_TOUCH
123     if(AP_MODE == true){
124       tft.setTextSize(1);
125       tft.setTextColor(ILI9341_RED);
126       tft.println("");
127       tft.println("    Connection timeout!");
128       tft.println("    Creating an AP Wi-Fi access...");
129       tft.setTextColor(ILI9341_GREEN);
130       tft.println("");
131       tft.print("    IP: ");
132       tft.print(apIP[0]);tft.print(".");tft.print(apIP[1]);tft.print(
    ".");tft.print(apIP[2]);tft.print(".");tft.println(apIP[3]);
133       tft.setTextSize(2);
134       tft.println("");
135       tft.println("");
136       tft.println("");
137       tft.println("");
138       tft.println(" Press any key to start...");
139     } else {
140       tft.setTextSize(1);
141       tft.setTextColor(ILI9341_GREEN);
142       tft.println("");
143       tft.println("    Connected !");
144       tft.println("");
145       tft.print("    IP: ");
146       tft.print(WiFi.localIP()[0]);tft.print(".");tft.print(WiFi.loca
    lIP()[1]);tft.print(".");tft.print(WiFi.localIP()[2]);tft.print(".")
    ;tft.println(WiFi.localIP()[3]);
147       tft.println("");
148       tft.setTextSize(2);
149       tft.println("");
150       tft.println("");
151       tft.println("");
152       tft.println("");
153       tft.println(" Press any key to start...");
154     }
155   #endif
156
157     // WEB SERVER SOCKET REQUESTS //
158
159     // Reflow Temperatures Array Data
```

```
160     webServer.on("/reflowdata", []() {
161       webServer.send(200, "text/plain", profileplotprint() );
162     });
163
164     // Thermocouple Plain Read
165     webServer.on("/temp", []() {
166       webServer.send(200, "text/plain", thermocouple_temp() );
167     });
168
169     // Settings - Wi-Fi
170     webServer.on("/settings/wifi", []() {
171       webServer.send(200, "text/html",
    header+title+settingsmenu+wifisettings1+String(ssid)+wifisettings2+S
    tring(password)+wifisettings3+bottom+webend);
172     });
173
174     // Settings - Profile 1
175     webServer.on("/settings/profile/1", []() {
176       webServer.send(200, "text/html",
    header+title+settingsmenu+setprojava+profileset(1)+bottom+webend );
177     });
178
179     // Settings - Profile 2
180     webServer.on("/settings/profile/2", []() {
181       webServer.send(200, "text/html",
    header+title+settingsmenu+setprojava+profileset(2)+bottom+webend );
182     });
183
184     // Settings - Profile 3
185     webServer.on("/settings/profile/3", []() {
186       webServer.send(200, "text/html",
    header+title+settingsmenu+setprojava+profileset(3)+bottom+webend );
187     });
188
189     // Settings - Profile 4
190     webServer.on("/settings/profile/4", []() {
191       webServer.send(200, "text/html",
    header+title+settingsmenu+setprojava+profileset(4)+bottom+webend );
192     });
193
194     // Wi-Fi save config. with path arguments from URL
195     webServer.on("/wifi/{}/{}", []() {
196       String Argument = webServer.pathArg(0);
197       Argument.toCharArray(ssid, 32);
198       Argument = webServer.pathArg(1);
199       Argument.toCharArray(password, 32);
200       savewifi();
201       webServer.sendHeader("Location", "/settings/wifi");
202       webServer.send(302);  // Code 302 - Found - Resource requested
    has been temporarily moved to the URL given by the Location
203     });
204
205     // Set profile and return to home
206     webServer.on("/set/{}", []() {
207       String Argument = webServer.pathArg(0);
208       current_profile = Argument.toInt();
209       webServer.sendHeader("Location", "/");
210       webServer.send(302);
```

```
211     });
212
213     // Save Profile Data
214     webServer.on("/profile/{}/{}/{}/{}/{}/{}/{}/{}/{}/{}", []() {
215
216       String Argument = webServer.pathArg(0);
217       int pts = Argument.toInt();      // Profile Number
218
219       Argument = webServer.pathArg(1);  // Profile Name
220
221       Argument.replace("%20", " ");    // Replace URL Space char
222
223       switch (pts) {
224         case 1:
225           Argument.toCharArray(profile1, 32);
226           break;
227         case 2:
228           Argument.toCharArray(profile2, 32);
229           break;
230         case 3:
231           Argument.toCharArray(profile3, 32);
232           break;
233         case 4:
234           Argument.toCharArray(profile4, 32);
235           break;
236       }
237
238       // Temperature Arguments
239       Argument = webServer.pathArg(2);
240       profile_param[0+(8*(pts-1))]= Argument.toInt();
241       Argument = webServer.pathArg(3);
242       profile_param[1+(8*(pts-1))]= Argument.toInt();
243       Argument = webServer.pathArg(4);
244       profile_param[2+(8*(pts-1))]= Argument.toInt();
245       Argument = webServer.pathArg(5);
246       profile_param[3+(8*(pts-1))]= Argument.toInt();
247       // Time Arguments
248       Argument = webServer.pathArg(6);
249       profile_param[4+(8*(pts-1))]= Argument.toInt();
250       Argument = webServer.pathArg(7);
251       profile_param[5+(8*(pts-1))]= Argument.toInt();
252       Argument = webServer.pathArg(8);
253       profile_param[6+(8*(pts-1))]= Argument.toInt();
254       Argument = webServer.pathArg(9);
255       profile_param[7+(8*(pts-1))]= Argument.toInt();
256
257       saveprofile(pts);    // Save params to the EEPROM
258
259       switch (pts) {       // Web page redirect
260         case 1:
261           webServer.sendHeader("Location", "/settings/profile/1");
262           break;
263         case 2:
264           webServer.sendHeader("Location", "/settings/profile/2");
265           break;
266         case 3:
267           webServer.sendHeader("Location", "/settings/profile/3");
268           break;
```

```
269         case 4:
270           webServer.sendHeader("Location", "/settings/profile/4");
271           break;
272       }
273       webServer.send(302);
274     });
275
276     // Reboot the board
277     webServer.on("/reboot", []() {
278       webServer.send(200, "text/html", "Restarting... please wait...
    <a style=\"color: teal;\" href=\"/\">Home Page</a>"+jsreboot );
279       ESP.restart();
280     });
281
282     // Start Reflow
283     webServer.on("/start", []() {
284       webServer.sendHeader("Location", "/");
285       webServer.send(302);
286       savecurrentprofile(current_profile);  // Save Current Profile
287       REFLOW_STATUS = true;
288     });
289
290     // Stop Reflow
291     webServer.on("/stop", []() {
292       webServer.sendHeader("Location", "/");
293       webServer.send(302);
294       REFLOW_STATUS = false;
295     });
296
297     // Replay to all not defined requests with same home HTML file
298     webServer.onNotFound([]() {
299       if(AP_MODE == true){
300       webServer.send(200, "text/html",
    header+title+menu(apIP[0],apIP[1],apIP[2],apIP[3])+mainhome()+bottom
    +profiledata(current_profile)+mainscripts+webend );
301       } else {
302       webServer.send(200, "text/html",
    header+title+menu(WiFi.localIP()[0], WiFi.localIP()[1],
    WiFi.localIP()[2],
    WiFi.localIP()[3])+mainhome()+bottom+profiledata(current_profile)+ma
    inscripts+webend );
303       }
304     });
305
306     webServer.begin(); // Starts the previous defined Web Server
307
    308     // Starts the loop0() on CORE 0 (By default the main loop()
    is running on CORE 1)
309     xTaskCreatePinnedToCore(loop0,"loop0",10000,NULL,1,&Tloop0,0);
     /* pin task to core 0 */
310
311     digitalWrite(STATUS_LED, HIGH);
312
313   }
314
315   // WEB SERVER // CORE 1 //
316   void loop() {
317     // DNS Server for Wi-Fi in AP MODE
```

```
318     if(AP_MODE == true){dnsServer.processNextRequest();}
319     // Web Server Client Handler
320     webServer.handleClient();
321   }
322
323   // CONTROL // LCD // CORE 0 //
324   void loop0( void * pvParameters ){
325    while(1){
326
327     currentMillis = millis();
328
329     if(REFLOW_STATUS == true){
330
331       if(prev_REFLOW_STATUS == false){    // Reflow initialized
332         temps_pos = 0;
333         temp_acqui = thermocouple.readCelsius();
334         prev_temp_acqui = temp_acqui;
335         Degsec = 0 ;
336         digitalWrite(SSR_PIN, HIGH);
337         reflow_pos = 1;
338
339         reflow_wait = currentMillis;
340         plot_flag = currentMillis;
341         acquisition_flag = currentMillis;
342         control_flag = currentMillis;
343         PWM_flag = currentMillis;
344         prev_REFLOW_STATUS = true;
345       }
346
347
348       // Oven Temperature acquisition ( every 300 ms ) ( 10 x 2 sec )
349       if (currentMillis - acquisition_flag >= 300)
350       {
351         if( temps_pos >= 10 ){
352           Degsec = ((temp_acqui - prev_temp_acqui)*100) / 3 ;
353           prev_temp_acqui = temp_acqui;
354           temps_pos = 0 ;
355         }
356         int tempstemp = thermocouple.readCelsius();
357         if( ( (temp_acqui-tempstemp) > -25) && ( (temp_acqui-
    tempstemp) < 25) ){
358             temp_acqui = (temp_acqui + tempstemp) / 2 ;
359         }
360         ++temps_pos;
361         acquisition_flag = currentMillis;
362       }
363
364       // Oven PWM Power Controller  ( variable )
365       if(currentMillis - PWM_flag >= 1000){
366
367         if(PWM_period > 20){
368           digitalWrite(SSR_PIN, HIGH);
369         }
370
371         PWM_flag = currentMillis;
372
373       }else if(currentMillis - PWM_flag >= PWM_period){
374
```

```
375         if(PWM_period < 980){
376           digitalWrite(SSR_PIN, LOW);
377         }
378
379       }
380
381
382       // Oven Power Controller  ( every 1000 ms )
383       if (currentMillis - control_flag >= 1000)
384       {
385
386         PWM_period = PWM_period + ((ramp(current_profile,reflow_pos)-
    Degsec));
387
388           if ( PWM_period > 1000 ){PWM_period=1000;}
389           if ( PWM_period < 0 ){PWM_period=0;}
390
391         if( ramp(current_profile,reflow_pos) == 0 ){
392
393             if (currentMillis - reflow_wait >=
    maxtime(current_profile,reflow_pos) ){
394                 ++reflow_pos;
395                 reflow_wait = currentMillis;
396             }
397
398         }else{
399
400           if( maxtempoint(current_profile,reflow_pos) <=
    (temp_acqui+15) ){
401             ++reflow_pos;
402             reflow_wait = currentMillis;
403           }
404         }
405         if(reflow_pos >= 5){REFLOW_STATUS=false;}
406         control_flag = currentMillis;
407
408       }
409
410
411       // Oven Live Plot ( every 5000 ms )
412       if ((currentMillis - plot_flag >= 5000) && (plotleg < 138))
413       {
414         plotdata[2*plotleg] = temp_acqui;
415         plotdata[2*plotleg+1] = plotleg*5;
416         plotleg=plotleg+1;
417         plot_flag = currentMillis;
418       }
419
420     }else{  // REFLOW_STATUS != true
421         digitalWrite(SSR_PIN, LOW);       // Disable Power
422         prev_REFLOW_STATUS = false;
423     }  // ENF IF REFLOW_STATUS
424
425     // NTC safety temperature tester
426     if (currentMillis - NTC_flag >= 1500){
427
428       NTC_temp = (3.3/4095)*analogRead(4);    // Voltage value
429       NTC_temp = Rsec/((3.3/NTC_temp)-1);     // Resistance NTC
```

```
430       NTC_temp = (NTCbeta/(log(NTC_temp/NTCR0)+(NTCbeta/298)))-273;
      // Temp NTC
431
432       if(NTC_temp > 80 && REFLOW_STATUS == true){
433         REFLOW_STATUS=false;     // Disable de reflow
434          ledcWriteTone(0, 2000);
435          ledcWrite(0, 125);
436          for (int i = 0; i <= 5; i++) { // Beep to warn of the fault
437             ledcWrite(0, 125);
438             delay(500);
439             ledcWrite(0, 0);
440             delay(500);
441          }
442          ledcWrite(0, 0);
443       }
444     }
445
446   #ifdef LCD_TOUCH
447
448     // LCD INI
449     if(TouchRead == true && LCD_screen == 0){
450       LCDPrintProfiles(false);
451       prev_LCD_profile = current_profile;
452       LCDPrintTemp(false,0);
453       LCD_screen = 1 ;
454       LCD_temp_flag = currentMillis;
455     }
456
457     // Touch Read
458     if(currentMillis - touch_flag >= 250){
459
460         // Get the currently touched pads
461         currtouched = touch.touched();
462
463         for (uint8_t i=0; i<12; i++) {
464           // it if *is* touched and *wasnt* touched before, alert!
465           if ((currtouched & _BV(i)) && !(lasttouched & _BV(i)) ) {
466            #ifdef SERIAL_DEBUG
467              Serial.print(i); Serial.println(" touched");
468             #endif
469
470               if(TouchRead == false && LCD_screen == 1 &&
      REFLOW_STATUS == false){
471
472               if(i == 1){current_profile=current_profile-1;}  // [+]
473               if(i == 0){current_profile=current_profile+1;}  // [-]
474
475               if(current_profile < 1){current_profile = 1;}
476               if(current_profile > 4){current_profile = 4;}
477
478             }
479
480             // [Start] Button
481             if(LCD_screen == 1 && i == 4 && REFLOW_STATUS == false){
482               timeoutstart = currentMillis;
483             }
484
485             // [Stop] Button
```

```
486             if(LCD_screen == 1 && i == 3 && REFLOW_STATUS == true){
487               timeoutstop = currentMillis;
488             }
489
490             TouchRead = true;
491           }
492           // if it *was* touched and now *isnt*, alert!
493           if (!(currtouched & _BV(i)) && (lasttouched & _BV(i)) ) {
494            #ifdef SERIAL_DEBUG
495             Serial.print(i); Serial.println(" released");
496             #endif
497             TouchRead = false;
498             timeoutstart = 0;
499             timeoutstop = 0;
500           }
501         }
502
503         if(prev_LCD_profile != current_profile && LCD_screen == 1 &&
    REFLOW_STATUS == false){
504           LCDChangeProfiles(current_profile,prev_LCD_profile);
505         }
506
507         prev_LCD_profile = current_profile;
508
509         // [Start] Hold Time Button
510         if((timeoutstart > 0) && (currentMillis - timeoutstart >=
    2500)){
511             REFLOW_STATUS = true;
512             savecurrentprofile(current_profile);
513         }
514         // [Stop] Hold Time Button
515         if((timeoutstop > 0) && (currentMillis - timeoutstop >=
    2500)){
516             REFLOW_STATUS = false;
517         }
518
519         if(prev_RFLW != REFLOW_STATUS && LCD_screen == 1){
520           LCDReflowStatus(REFLOW_STATUS);
521         }
522
523         prev_RFLW = REFLOW_STATUS;
524
525         // reset our state
526         lasttouched = currtouched;
527
528         touch_flag = currentMillis;
529     }
530
531     // Termocoupler Read
532     if((LCDtempread == true) && (LCD_screen == 1) && (currentMillis -
    LCD_temp_flag >= 750)){
533
534       temp_LCD = thermocouple.readCelsius();
535       if(temp_LCD > 1000){temp_LCD = -1;}
536       LCDtempread = false;
537
538     // LCD Print bottom Temperature
```

```
539    }else if((LCD_screen == 1) && (currentMillis - LCD_temp_flag >=
     1500)){
540        if(prev_temp_LCD != temp_LCD){
541          LCDPrintTemp(true,temp_LCD);
542        }
543        LCDtempread = true;
544        prev_temp_LCD = temp_LCD;
545        LCD_temp_flag = currentMillis;
546      }// END LCD
547
548   #endif
549
550       vTaskDelay(10);    // Watchdog trigger fix
551     } // END while(1)
552   } // END loop0()
```

B2. Variables.h

```
01   // PIN DEFINITIONS //
02   #define SSR_PIN 17
03   #define STATUS_LED 33
04   // SPI LCD PINS
05   #define TFT_DC 27
06   #define TFT_CS 26
07   #define TFT_MOSI 23
08   #define TFT_CLK 18
09   #define TFT_RST 14
10   #define TFT_MISO 19
11   // MAX6675 Pins
12   #define thermoDO 19
13   #define thermoCS 5
14   #define thermoCLK 18
15
16   // Time Out for AP MODE
17   const int      conn_time_out = 10000 ;      // in ms
18   int            time_out = 0 ;
19
20   // Server Settings //
21   const byte     DNS_PORT = 53;          // Capture DNS requests on port
53
22   IPAddress      apIP(192, 168, 4, 1); // Network for server in AP MODE
23   DNSServer      dnsServer;            // Create the DNS object
24   WebServer      webServer(80);        // HTTP server
25   boolean        AP_MODE = false;
26
27   // Dual Core Control loop on CORE 0
28   TaskHandle_t  Tloop0;
29   boolean      REFLOW_STATUS = false; // Reflow process status (on/off)
30   boolean        prev_REFLOW_STATUS = false;
31
32   // No delay variables
33   unsigned long currentMillis;
34   unsigned long control_flag;
35   unsigned long acquisition_flag;
36   unsigned long plot_flag;
37   unsigned long PWM_flag;
38   unsigned long reflow_wait;
39   unsigned long NTC_flag;
40   long PWM_period = 1000;
41
42   // Reflow Control variables
43   int temps_pos, temp_acqui, prev_temp_acqui, Degsec, reflow_pos;
44
45   // Termistor NTC variables
46   #define Rsec 1500
47   #define NTCbeta 3977
48   #define NTCR0 10000
49   float NTC_temp = 0.0;
50
51   // Live Plot Data Arrays
52   int plotdata[276];
53   int plotleg=0;
```

```
54
55   // LCD CUSTOM COLORS
56   #define LIGHT_YELLOW 0xFFF0
57   #define LIGHT_GRAY 0xDEFB
58   #define DARK_RED 0x9800
```

B3. LCD.h

```
001   // LCD Variables
002   int LCD_screen = 0;
003   int temp_LCD,prev_temp_LCD,prev_LCD_profile;
004   boolean LCDtempread = false;
005   boolean prev_RFLW = false;
006   unsigned long LCD_temp_flag;
007
008   // Touch Variables
009   #ifndef _BV
010   #define _BV(bit) (1 << (bit))
011   #endif
012
013   unsigned long touch_flag;
014   unsigned long timeoutstart = 0;
015   unsigned long timeoutstop = 0;
016   boolean TouchRead = false ;
017
018   // Keeps track of the last pins touched
019   // so we know when buttons are 'released'
020   uint16_t lasttouched = 0;
021   uint16_t currtouched = 0;
022
023   // TFT Print Profiles
024   void LCDPrintProfiles(boolean initz) {
025     if(initz == false){
026       tft.fillScreen(ILI9341_BLACK);
027       tft.fillScreen(ILI9341_BLACK);
028       tft.setTextSize(2);
029       tft.setTextColor(ILI9341_BLACK);
030       tft.fillRect(0, 0, 320, 18, ILI9341_BLUE);
031       tft.setCursor(2, 2);
032       tft.println("Reflow Profil List");
033     }else{
034       tft.fillRect(0, 19, 320, 80, ILI9341_BLACK);
035       tft.fillRect(0, 19, 320, 80, ILI9341_BLACK);
036     }
037     String txt = String(profile1);
038     ///Print current_profile  in LIGHT_YELLOW
039     for (int i = 1; i <= 4; i++) {
040
041       if(current_profile == i){
042         tft.fillRect(0,20*i,320,18, LIGHT_YELLOW);
043         tft.setTextColor(ILI9341_BLACK);
044       }else{
045         tft.setTextColor(ILI9341_BLUE);
046       }
047
048       tft.setCursor(2,(20*i)+2);
049       switch (i) {
050           case 1:
051             //txt = String(profile1);
052             txt.remove(26, 5);
053             tft.println(txt);
054             break;
055           case 2:
```

```
056             txt = String(profile2);
057             txt.remove(26, 5);
058             tft.println(txt);
059             break;
060           case 3:
061             txt = String(profile3);
062             txt.remove(26, 5);
063             tft.println(txt);
064             break;
065           case 4:
066             txt = String(profile4);
067             txt.remove(26, 5);
068             tft.println(txt);
069             break;
070       }
071     } // END FOR
072   }
073
074   // TFT Profiles Change
075   void LCDChangeProfiles(int current, int prev){
076     String txt = String(profile1);
077     int set = prev;
078     for(int i = 0; i <= 1; i++){
079
080       if(i==0){
081           tft.fillRect(0, (20*prev)-1, 320, 21, ILI9341_BLACK);
082           tft.fillRect(0, (20*prev)-1, 320, 21, ILI9341_BLACK);
083           tft.setTextColor(ILI9341_BLUE);
084           tft.setCursor(2,(20*prev)+2);
085
086       }else if(i==1){
087           tft.fillRect(0,20*current,320,18, LIGHT_YELLOW);
088           tft.setTextColor(ILI9341_BLACK);
089           tft.setCursor(2,(20*current)+2);
090           set = current;
091       }
092     switch (set) {
093       case 1:
094            //txt = String(profile1);
095            txt.remove(26, 5);
096            tft.println(txt);
097            break;
098       case 2:
099            txt = String(profile2);
100           txt.remove(26, 5);
101           tft.println(txt);
102           break;
103       case 3:
104           txt = String(profile3);
105           txt.remove(26, 5);
106           tft.println(txt);
107           break;
108       case 4:
109           txt = String(profile4);
110           txt.remove(26, 5);
111           tft.println(txt);
112           break;
113     }
```

```
114     }// END FOR
115   }
116
117   // LCD Print Bottom Temperatura
118   void LCDPrintTemp(boolean initz, int tempread){
119     if(initz == false){
120       tft.fillRect(0, 221, 320, 19, LIGHT_GRAY);
121       tft.setTextSize(2);
122       tft.setCursor(2, 223);
123       tft.setTextColor(DARK_RED);
124       tft.print("Temperature:");
125     }else if(tempread < 0){
126       tft.fillRect(145, 221, 320, 19, LIGHT_GRAY);
127       tft.fillRect(145, 221, 320, 19, LIGHT_GRAY);
128       tft.setTextSize(2);
129       tft.setCursor(146, 223);
130       tft.setTextColor(DARK_RED);
131       tft.print("Not Connected");
132     }else if(initz == true){
133       tft.fillRect(145, 221, 320, 19, LIGHT_GRAY);
134       tft.fillRect(145, 221, 320, 19, LIGHT_GRAY);
135       tft.setTextSize(2);
136       tft.setCursor(146, 223);
137       tft.setTextColor(DARK_RED);
138       tft.print(tempread);
139       tft.print((char)247);
140       tft.print("C");
141     }
142   }
143
144   // LCD Print Reflow Status On/Off
145   void LCDReflowStatus(boolean initz){
146     if(initz == true){
147       tft.fillRect(294, 0, 320, 18, ILI9341_BLUE);
148       tft.fillRect(294, 0, 320, 18, ILI9341_BLUE);
149       tft.setTextSize(2);
150       tft.setCursor(295, 2);
151       tft.setTextColor(ILI9341_GREEN);
152       tft.print("ON");
153     }else if(initz == false){
154       tft.fillRect(294, 0, 320, 18, ILI9341_BLUE);
155       tft.fillRect(294, 0, 320, 18, ILI9341_BLUE);
156     }
157   }
```

B4. EEPROM.h

```
001   //  EEPROM 512 BYTES
002   //  ----------------------------
003   //  32  SSID          { NAMES }
004   //  32  PASSWORD
005   //  32  Profile 1
006   //  32  Profile 2
007   //  32  Profile 3
008   //  32  Profile 4
009   //  + ------ > 192 bytes for names
010   //  128 bytes for 4 Byte params[]
011   //  + ------ > 320 bytes
012   //  4 bytes actual profile
013
014   // WiFi Settings
015   char ssid[32];
016   char password[32];
017
018   // Profile Names
019   char profile1[32];
020   char profile2[32];
021   char profile3[32];
022   char profile4[32];
023
024   // 4 Profiles * ( 4 Temps ; 4 Time )
025   int profile_param[32];
026
027   // Current Selected Profile
028   int current_profile;
029
030   // Load all EEPROM data
031   void loadEEPROMdata() {
032     EEPROM.begin(512);
033     // NAMES
034     EEPROM.get(0, ssid);
035     EEPROM.get(32, password);
036     EEPROM.get(64, profile1);
037     EEPROM.get(96, profile2);
038     EEPROM.get(128, profile3);
039     EEPROM.get(160, profile4);
040     // Profile Params
041     for (int i = 0; i < 32; i = i + 1) {
042      EEPROM.get(192+(i*4), profile_param[i]);
043     }
044     EEPROM.get(400, current_profile);
045     EEPROM.end();
046   }
047
048   // Save Wi-Fi Credentials
049   void savewifi() {
050     EEPROM.begin(512);
051     EEPROM.put(0, ssid);
052     EEPROM.put(32, password);
053     EEPROM.commit();
054     EEPROM.end();
055   }
```

```
056
057   // Save Profile Settings
058   void saveprofile(int num){
059     EEPROM.begin(512);
060     switch (num) {
061       case 1:
062         EEPROM.put(64, profile1);
063         break;
064       case 2:
065         EEPROM.put(96, profile2);
066         break;
067       case 3:
068         EEPROM.put(128, profile3);
069         break;
070       case 4:
071         EEPROM.put(160, profile4);
072         break;
073     }
074     // Profile Params
075     for (int i = (0 + 8*(num-1) ) ; i < 8*num ; i = i + 1) {
076      EEPROM.put(192+(i*4), profile_param[i]);
077     }
078     EEPROM.commit();
079     EEPROM.end();
080   }
081
082   // Save Current Selected Profile
083   void savecurrentprofile(int num){
084     EEPROM.begin(512);
085     EEPROM.put(400, num);
086     EEPROM.commit();
087     EEPROM.end();
088   }
089
090   int ramp(int profile,int pos){
091
092     int profileramp = 0;
093
094     switch (pos) {
095       case 1:
096         profileramp = ((profile_param[(0+8*(profile-1))]-
      25)*100)/profile_param[(4+8*(profile-1))];
097         break;
098       case 2:
099         profileramp = ((profile_param[(1+8*(profile-1))]-
      profile_param[(0+8*(profile-
      1))])*100)/(profile_param[(5+8*(profile-1))]-
      profile_param[(4+8*(profile-1))]);
100         break;
101       case 3:
102         profileramp = ((profile_param[(2+8*(profile-1))]-
      profile_param[(1+8*(profile-
      1))])*100)/(profile_param[(6+8*(profile-1))]-
      profile_param[(5+8*(profile-1))]);
103         break;
104       case 4:
```

```
105         profileramp = ((profile_param[(3+8*(profile-1))]-
    profile_param[(2+8*(profile-1))])*100)/(profile_param[(7+8*(profile-
    1))]-profile_param[(6+8*(profile-1))]);
106         break;
107       }
108         return profileramp;
109     }
110
111     int maxtempoint(int profile,int pos){
112
113       int returnvalue = 0;
114
115       switch (pos) {
116         case 1:
117           returnvalue = profile_param[(0+8*(profile-1))];
118           break;
119         case 2:
120           returnvalue = profile_param[(1+8*(profile-1))];
121           break;
122         case 3:
123           returnvalue = profile_param[(2+8*(profile-1))];
124           break;
125         case 4:
126           returnvalue = profile_param[(3+8*(profile-1))];
127           break;
128       }
129         return returnvalue;
130     }
131
132     int maxtime(int profile,int pos){
133
134       int returnvalue = 0;
135
136       switch (pos) {
137         case 1:
138           returnvalue = profile_param[(4+8*(profile-1))]*1000;
139           break;
140         case 2:
141           returnvalue = (profile_param[(5+8*(profile-1))]-
    profile_param[(4+8*(profile-1))])*1000;
142           break;
143         case 3:
144           returnvalue = (profile_param[(6+8*(profile-1))]-
    profile_param[(5+8*(profile-1))])*1000;
145           break;
146         case 4:
147           returnvalue = (profile_param[(7+8*(profile-1))]-
    profile_param[(6+8*(profile-1))])*1000;
148           break;
149       }
150         return returnvalue;
151     }
```

B5. HTML.h

```
001   // WEB MAIN PAGE HEADER /w STYLE CSS //
002   static const String header  = "<!DOCTYPE html><html><head>"
003     "<title>Reflow Control Panel</title><meta name=\"viewport\"
      content=\"width=device-width, initial-scale=1.0\">"
004     "<link rel=\"shortcut icon\"
      href=\"https://www.jrodrigo.net/files/favicon.ico\">"

005     "<style>"
006         "*{box-sizing: border-box;}"
007         ".menu{"
008         "float: left;"
009         "width: 30%;}"
010         ".menuitem{"
011         "padding: 8px;"
012         "margin-top: 7px;"
013         "border-bottom: 1px solid #f1f1f1;}"
014         ".main{"
015         "float: left;"
016         "width: 70%;"
017         "padding: 0 20px;"
018         "overflow: hidden;}"
019       "graph{"
020               "margin: 10px auto;"
021               "border-style: solid;"
022               "border-width: 1px;"
023               "width: 650px;"
024               "height: 400px;}"
025       "@media only screen and (max-width:800px){"
026         ".main{"
027         "width: 80%;"
028         "padding: 0;}"
029         ".right{"
030         "width: 100%;"
031       "}}"
032       "@media only screen and (max-width:500px){"
033         ".menu, .main, .right{"
034         "width: 100%;}}"
035     "</style>"
036     "</head><body style=\"font-family:Verdana;\">";
037
038   // WEB MAIN TOP TITLE //
039   static const String title = "<div style=\"background-
      color:#f1f1f1;padding:15px;\"><h2>Reflow Control Panel</h2></div>";
040
041   // WEB HOME - MAIN LEFT MENU //
042   String menu(int IP1, int IP2, int IP3, int IP4){
043
044       String menu  = "<div style=\"overflow:auto\"><div
      class=\"menu\">"
045                       "<div class=\"menuitem\">Current Profile
      <select onchange=\"javascript:location.href = this.value;\"";
046
```

```
047                     if(REFLOW_STATUS == true){ menu += " disabled"; }
048
049             menu  += "><option value=\"/set/1\"";
050
051                     if(current_profile == 1){ menu += " selected"; }
052
053             menu  += ">1 : "+ String(profile1)+"</option><option
    value=\"/set/2\"";
054
055                     if(current_profile == 2){ menu += " selected"; }
056
057
058             menu  += ">2 : "+ String(profile2)+"</option><option
    value=\"/set/3\"";
059
060                      if(current_profile == 3){ menu += " selected"; }
061
062             menu  += ">3 : "+ String(profile3)+"</option><option
    value=\"/set/4\"";
063
064                      if(current_profile == 4){ menu += " selected"; }
065
066             menu  += ">4 : "+ String(profile4)+"</option>"
067
068                       "</select></div>"
069                       "<div class=\"menuitem\">Current Temp.: <span
    id=\"temp\">nan &deg;C</span></div>"
070                       "<div class=\"menuitem\">Current
    IP:  "+String(IP1)+"."+String(IP2)+"."+String(IP3)+"."+String(IP4)+
    "</div>"
071                       "<div class=\"menuitem\"> </div>";
072
073            if(REFLOW_STATUS == false){
074             menu  += "<div class=\"menuitem\"><input type=\"submit\"
    value=\"Start Reflow\"
    onclick=\"window.location.href='/start'\"></div>";
075             } else {
076             menu  += "<div class=\"menuitem\"><input type=\"submit\"
    value=\"STOP Reflow\"
    onclick=\"window.location.href='/stop'\"></div>";
077             }
078
079             menu  += "<div class=\"menuitem\"> </div>"
080                       "<div class=\"menuitem\"><a
    href=\"/settings/profile/1\">Settings</a></div>"
081                       "</div>";
082
083       return menu;
084   }
085
086   // WEB HOME - CANVAS ZONE //
087   String mainhome(){
088
089     String mainhome  = "<div class=\"main\"><h3>Profile ";
090
091        switch (current_profile) {
092           case 1:
093             mainhome += "1 : "+String(profile1);
```

```
094             break;
095           case 2:
096             mainhome += "2 : "+String(profile2);
097             break;
098           case 3:
099             mainhome += "3 : "+String(profile3);
100             break;
101           case 4:
102             mainhome += "4 : "+String(profile4);
103             break;
104       }
105
106     mainhome += "</h3><p><canvas
    id=\"graph\"></canvas></p></div></div>";
107     return mainhome;
108   }
109
110   // Live Canvas Graphing Scripts //
111   static const String mainscripts = "<script
    src=\"data:text/javascript; base64,Ly8gVXBkYXRlIHRoZXJtb2NvdXBsZSB0ZW
    1wCnZhciB0ZW1wdGltbWVyID0gc2V0SW50ZXJ2YWwoZnVuY3Rpb24oKSB7CiAgdmFyIH
    hodHRwID0gbmV3IFhNTEh0dHBSZXF1ZXN0KCk7CiAgeGh0dHAub25yZWFkeXN0YXRlY2
    hhbmdlID0gZnVuY3Rpb24oKSB7CiAgICBpZiAodGhpcy5yZWFkeVN0YXRlID09IDQgJi
    YgdGhpcy5zdGF0dXMgPT0gMjAwKSB7CiAgICAgIGRvY3VtZW50LmdldEVsZW1lbnRCeU
    lkKCJ0ZW1wIikuaW5uZXJIVE1MID0KICAgICAgdGhpcy5yZXNwb25zZVRleHQgKyAiIC
    ZkZWc7QyI7CiAgICB9CiAgfTsKICB4aHR0cC5vcGVuKCJHRVQiLCAiL3RlbXAiLCB0cn
    VlKTsKICB4aHR0cC5zZW5kKCk7Cn0sIDI1NTEpOwoKdmFyIGdyYXBoID0gZG9jdW1lbn
    QuZ2V0RWxlbWVudEJ5SWQoJ2dyYXBoJyk7CnZhciBjb250ZXh0ID0gZ3JhcGguZ2V0Q2
    9udGV4dCgnMmQnKTsKdmFyCXdpZHRoID0gZ3JhcGgud2lkdGggPSA2NTY7CnZhcgloZW
    lnaHQgPSBncmFwaC5oZWlnaHQgPSA0MjU7CmNvbnRleHQuZmlsbFN0eWxlID0gJyNmNm
    Y2ZjYnOwpjb250ZXh0LmZpbGxSZWN0KDAsIDAsIHdpZHRoLCBoZWlnaHQpOwpjb250ZX
    h0LmZpbGxTdHlsZSA9ICIjMDAwMDAwIjsKY29udGV4dC5mb250ID0gIjEycHggQXJpYW
    wiOwp2YXIgaSA9IDI1Owpmb3IoOyBpID4gewoJLy8gVGVtcCBWYWx1ZQoJY29udGV4dC5maWxsVGV4dChpLDEsaGVpZ2h0LWkpLy8gTGluZWFzIEhvcml6b2
    50YWxlcwoJY29udGV4dC5iZWdpblBhdGgoKTsKCWNvbnRleHQubW92ZVRvKDIyLCBoZW
    lnaHQtaS00KTsJIC8vIFhpLFlpCgljb250ZXh0LmxpbmVUbyh3aWR0aC0zMCwgaGVpZ2
    h0LWktNCk7IC8vIFhmLFlmCgljb250ZXh0LmxpbmVXaWR0aCA9IDAuMjU7Cgljb250ZX
    h0LmxpbmVDYXAgPSAncm91bQnOwoJY29udGV4dC5zdHJva2VTdHlsZSA9ICcjM2IzYj
    NiJzsKCWNvbnRleHQuc3Ryb2tlKCk7CglpID0gaSsyNTsKfQpjb250ZXh0LmZvbnQgPS
    AiOXB4IEFyaWFsIjsKY29udGV4dC5maWxsVGV4dChTdHJpbmcuZnJvbUNoYXJDb2RlKD
    E4NikrIkMiLDExLDEwKTsKY29udGV4dC5maWxsU3R5bGUgPSAiIzAwMDAwMCI7CmNvbn
    RleHQuZm9udCA9ICIxMnB4IEFyaWFsIjsKY29udGV4dC50ZXh0QWxpZ24gPSAiY2VudG
    VyIjsKdmFyIGkgPSAwOwpmb3IoOyBpIDwgKHdpZHRoLTMwKTspIHsKCS8vIFRlbXAgVm
    FsdWUKCWNvbnRleHQuZmlsbFRleHQoaSwyNStpLGhlaWdodC0xMCk7CgkKCS8vIExpbm
    VhcyBIb3Jpem9udGFsZXMKCWNvbnRleHQuYmVnaW5QYXRoKCk7Cgljb250ZXh0Lm1vdm
    VUbyhpKzI1LCAyMCk7CSAvLyBYaSxZaQoJY29udGV4dC5saW5lVG8oaSsyNSwgaGVpZ2
    h0LTI1KTsgLy8gWGYsWWYKCWNvbnRleHQubGluZVdpZHRoID0gMC4yNTsKCWNvbnRleH
    QubGluZUNhcCA9ICdyb3VuZCc7Cgljb250ZXh0LnN0cm9rZVN0eWxlID0gJyMzYjNiM2
    InOwoJY29udGV4dC5zdHJva2UoKTsKCWkgPSBpKzMwOwp9CmNvbnRleHQuZm9udCA9IC
    I5cHggQXJpYWwiOwpjb250ZXh0LmZpbGxUZXh0KCJzZWMiLHdpZHRoLTEyLGhlaWdodC
    0xMCk7CmNvbnRleHQudHJhbnNsYXRlKDI1LCAtMjUpOwovLyBQcmludCBSZWZsb3cgUH
    JvZmlsZQp2YXIgcHJldl9zdGF0ID0gdGVtcHNbMF0sCglwcmV2X2xlZnQgPSB0aW1lc1
    swXTsKZm9yKHN0YXQgaW4gdGVtcHMpIHsKCXRoZV9zdGF0ID0gdGVtcHNbc3RhdF07Cg
    l0aGVfdGltZSA9IHRpbWVzW3N0YXRdCgljb250ZXh0LmJlZ2luUGF0aCgpOwoJY29udG
    V4dC5tb3ZlVG8ocHJldl9sZWZ0LCBoZWlnaHQtcHJldl9zdGF0KzIxKTsJIC8vIFhpLF
    lpCgljb250ZXh0LmxpbmVUbyh0aGVfdGltZSwgaGVpZ2h0LXRoZV9zdGF0KzIxKTsgLy
    8gWGYsWWYKCWNvbnRleHQubGluZVdpZHRoID0gMjsKCWNvbnRleHQubGluZUNhcCA9IC
    dyb3VuZCc7Cgljb250ZXh0LnN0cm9rZSgpOwoJcHJldl9zdGF0ID0gdGhlX3N0YXQ7Cg
```

```
    lwcmV2X2xlZnQgPSB0aGVfdGltZTsKfQoKCnZhciB4ID0gMTsKdmFyIHBsb3RkYXRhID
    0gWzBdOwp2YXIgcGxvdHRpbW1lciA9IHNldEludGVydmFsKGZ1bmN0aW9uKCkgewoKIC
    AgICB2YXIgeG1saHR0cHBsb3Q9bmV3IFhNTEh0dHBSZXF1ZXN0KCk7CiAgICAgeG1saH
    R0cHBsb3Qub25yZWFkeXN0YXRlY2hhbmdlPWZ1bmN0aW9uKCl7CiAgICAgICAgIGlmIC
    h4bWxodHRwcGxvdC5yZWFkeVN0YXRlPT00ICYmIHhtbGh0dHBwbG90LnN0YXR1cz09Mj
    AwKXsKICAgICAgICAgICAgIAoJCQkgcGxvdGRhdGEgPSB4bWxodHRwcGxvdC5yZXNwb2
    5zZVRleHQuc3BsaXQoIiwiKTsKCQkJCQkJCQkJCkCQkJIAlmb3IoOyB4IDwgKHBsb3RkYX
    RhLmxlbmd0aC8yKTsgeCsrKSB7CgkJCQkJCQljb250ZXh0LmJlZ2luUGF0aCgpOw
    oJCQkJCWNvbnRleHQubW92ZVRvKHBsb3RkYXRhW3gqMisxXSxoZWlnaHQtcGxvdGRhdG
    FbeCoyXSsyMCk7CSAvLyBYaSxZaQoJCQkJCWNvbnRleHQubGluZVRvKHBsb3RkYXRhWy
    h4LTEpKjIrMV0sIGhlaWdodC1wbG90ZGF0YVsoeC0xKSoyXSsyMCk7IAkvLyBYZixZZg
    oJCQkJCWNvbnRleHQubGluZVdpZHRoID0gMTsKCQkJCQljb250ZXh0LmxpbmVDYXAgPS
    Ancm91bmQnOwoJCQkJCWNvbnRleHQuc3Ryb2tlU3R5bGUgPSAnI2ZmMDAwMCc7CgkJCQ
    kJY29udGV4dC5zdHJva2UoKTsKCQkJCQkJCQkJCQkJfQkvLyBFTkQgRk9SCgkJSAKCQkJIA
    ogICAgICAgICB9CiAgICAgICAgfSAgCiAgICAgICAgICB9CiAgICAgICAgeG1saHR0cHBsb3Qub3BlbigiR0VUIiwiL3
    JlZmxvd2RhdGEiLHRydWUpOwogICAgIHhtbGh0dHBwbG90LnNlbmQoKTsKCQp9LCA0MT
    ExKTsgLy8gRU5EIFRJTU1FUg==\"></script>";
112   // WEB BOTTOM PROJECT PAGE DATA //
113   static const String bottom = "<div style=\"background-
    color:#f1f1f1;text-align:center;padding:10px;margin-top:7px;font-
    size:12px;\"> WiFi Reflow Controller - ESP32 - Version 0.0.1b - <a
    href=\"https://github.com/JRodrigoTech/ESP32-Reflow-Toaster-Oven-
    Controller\"  target=\"_blank\">GitHub</a></div>";
114
115   // WEB END CODE //
116   static const String webend = "</body></html>";
117
118   // SETTINGS - LEFT MENU //
119   static const String settingsmenu  = "<div
    style=\"overflow:auto\"><div class=\"menu\">"
120                                       "<div
    class=\"menuitem\"><h3>Settings</h3></div>"
121                                       "<div class=\"menuitem\"><a
    href=\"/settings/profile/1\">Profile 1</a></div>"
122                                       "<div class=\"menuitem\"><a
    href=\"/settings/profile/2\">Profile 2</a></div>"
123                                       "<div class=\"menuitem\"><a
    href=\"/settings/profile/3\">Profile 3</a></div>"
124                                       "<div class=\"menuitem\"><a
    href=\"/settings/profile/4\">Profile 4</a></div>"
125                                       "<div class=\"menuitem\"><a
    href=\"/settings/wifi\">Wi-Fi Connection</a></div>"
126                                       "<div class=\"menuitem\">
    </div>"
127                                       "<div class=\"menuitem\"><a
    href=\"/reboot\">Reboot</a></div>"
128                                       "<div class=\"menuitem\">
    </div>"
129                                       "<div class=\"menuitem\"><a
    href=\"/\">Back</a></div>"
130                                       "</div>";
131
132   // SETTINGS - Wi-Fi //
133   // Custom URL string call to save Wi-Fi credentials ::
    /wifi/{SSID}/{PASSWORD}
134   // and custom form with stored parameters
135   static const String wifisettings1 = "<script
    src=\"data:text/javascript; base64,ZnVuY3Rpb24gc2F2ZXdpZmkoKSB7CiAgIC
```

```
    B2YXIgc3NpZCA9IGRvY3VtZW50LmdldEVsZW1lbnRCeUlkKCJzc2lkIik7Cgl2YXIgcH
    dkID0gZG9jdW1lbnQuZ2V0RWxlbWVudEJ5SWQoInB3ZCIpOwkKICAgIHZhciBnb3VybC
    A9ICJodHRwOi8vIiArIHdpbmRvdy5sb2NhdGlvbi5ob3N0ICsgIi93aWZpLyIgKyBzc2
    lkLnZhbHVlICsgIi8iICsgcHdkLnZhbHVlOwogICAgd2luZG93LmxvY2F0aW9uID0gZ2
    91cmw7CQp9\"></script>"
136                                 "<div class=\"main\"><h3>WiFi
    Configuration</h3><div>SSID:<br><input type=\"text\" id=\"ssid\"
    value=\"";
137   static const String wifisettings2 = "\"><br>Password:<br><input
    type=\"text\" id=\"pwd\" value=\"";
138   static const String wifisettings3 =  "\"><br><br><input
    type=\"submit\" value=\"Save\"
    onClick=\"savewifi()\"></div></div></div>";
139
140   // SETTINGS - PROFILE //
141
142   // Custom URL string call to save profiles :: /profile/{Profile
    Num}/{Profile
    Name}/{Temp1}/{Temp2}/{Temp3}/{Temp4}/{Time1}/{Time2}/{Time3}/{Time4
    }/
143   static const String setprojava = "<script
    src=\"data:text/javascript; base64,ZnVuY3Rpb24gc2F2ZXByb2ZpbGUoKXsKIC
    AgIHZhciBuYW1lID0gZG9jdW1lbnQuZ2V0RWxlbWVudEJ5SWQoIm5hbWUiKTsKCXZhci
    Bwcm9maWxlbnVtID0gZG9jdW1lbnQuZ2V0RWxlbWVudEJ5SWQoInByb2ZpbGVudW0iKT
    sKCgl2YXIgdGVtcDEgPSBkb2N1bWVudC5nZXRFbGVtZW50QnlJZCgidGVtcDEiKTsJCg
    l2YXIgdGVtcDIgPSBkb2N1bWVudC5nZXRFbGVtZW50QnlJZCgidGVtcDIiKTsJCgl2YX
    IgdGVtcDMgPSBkb2N1bWVudC5nZXRFbGVtZW50QnlJZCgidGVtcDMiKTsJCgl2YXIgdG
    VtcDQgPSBkb2N1bWVudC5nZXRFbGVtZW50QnlJZCgidGVtcDQiKTsJCgoJdmFyIHRpbW
    UxID0gZG9jdW1lbnQuZ2V0RWxlbWVudEJ5SWQoInRpbWUxIik7CQoJdmFyIHRpbWUyID
    0gZG9jdW1lbnQuZ2V0RWxlbWVudEJ5SWQoInRpbWUyIik7Cgl2YXIgdGltZTMgPSBkb2
    N1bWVudC5nZXRFbGVtZW50QnlJZCgidGltZTMiKTsKCXZhciB0aW1lNCA9IGRvY3VtZW
    50LmdldEVsZW1lbnRCeUlkKCJ0aW1lNCIpOwoKICAgIHZhciBnb3VybCA9ICJodHRwOi
    8vIiArIHdpbmRvdy5sb2NhdGlvbi5ob3N0ICsgIi9wcm9maWxlLyIgKyBwcm9maWxlbn
    VtLnZhbHVlICsgIi8iICsgbmFtZS52YWx1ZSArIi8iICsgdGVtcDEudmFsdWUgKyAiLy
    8iICsgdGVtcDIudmFsdWUgKyAiLyIgKyB0ZW1wMy52YWx1ZSArIi8iICsgdGVtcDQudm
    FsdWUgKyAiLyIgKyB0aW1lMi52YWx1ZSArIi8iICsgdGltZTEudmFsdWUgKyAiLyIgKy
    BpbWUzLnZhbHVlICsgIi8iICsgdGltZTQudmFsdWU7CiAgICB3aW5kb3cubG9jYXRp
    Rpb24gPSBnb3VybDsJCn0=\"></script>";
144
145   // Profile Settings Form with EEPROM stored data
146   String profileset(int profilenum){
147
148       char profnam[32];
149       int values[8] = {0,0,0,0, 0,0,0,0};
150
151        switch (profilenum) {
152           case 1:
153             values[0] = profile_param[0];
154             values[1] = profile_param[1];
155             values[2] = profile_param[2];
156             values[3] = profile_param[3];
157             values[4] = profile_param[4];
158             values[5] = profile_param[5];
159             values[6] = profile_param[6];
160             values[7] = profile_param[7];
161             strcpy(profnam, profile1);
162             break;
163           case 2:
```

```
164             values[0] = profile_param[8];
165             values[1] = profile_param[9];
166             values[2] = profile_param[10];
167             values[3] = profile_param[11];
168             values[4] = profile_param[12];
169             values[5] = profile_param[13];
170             values[6] = profile_param[14];
171             values[7] = profile_param[15];
172             strcpy(profnam, profile2);
173             break;
174           case 3:
175             values[0] = profile_param[16];
176             values[1] = profile_param[17];
177             values[2] = profile_param[18];
178             values[3] = profile_param[19];
179             values[4] = profile_param[20];
180             values[5] = profile_param[21];
181             values[6] = profile_param[22];
182             values[7] = profile_param[23];
183             strcpy(profnam, profile3);
184             break;
185           case 4:
186             values[0] = profile_param[24];
187             values[1] = profile_param[25];
188             values[2] = profile_param[26];
189             values[3] = profile_param[27];
190             values[4] = profile_param[28];
191             values[5] = profile_param[29];
192             values[6] = profile_param[30];
193             values[7] = profile_param[31];
194             strcpy(profnam, profile4);
195             break;
196       }
197
198       String printhtml = "<div class=\"main\"><h3>Profile "
      +String(profilenum)+ " Configuration</h3>"
199                          "<div>Profile Name <input type=\"text\"
      id=\"name\" value=\"" +String(profnam)+ "\" maxlength=\"32\"><br>"
200                          "<input type=\"hidden\" id=\"profilenum\"
      value=\"" +String(profilenum)+ "\">"
201                          "Temperatures:<br>"
202
203                          "<input type=\"text\" id=\"temp1\" value=\""
      +String(values[0])+ "\">"
204                          "<input type=\"text\" id=\"temp2\" value=\""
      +String(values[1])+ "\">"
205                          "<input type=\"text\" id=\"temp3\" value=\""
      +String(values[2])+ "\">"
206                          "<input type=\"text\" id=\"temp4\" value=\""
      +String(values[3])+ "\">"
207
208                          "<br>Time:<br>"
209
210                          "<input type=\"text\" id=\"time1\" value=\""
      +String(values[4])+ "\">"
211                          "<input type=\"text\" id=\"time2\" value=\""
      +String(values[5])+ "\">"
```

```
212                         "<input type=\"text\" id=\"time3\" value=\""
    +String(values[6])+ "\">"
213                         "<input type=\"text\" id=\"time4\" value=\""
    +String(values[7])+ "\">"
214
215
216                         "<br><br><input type=\"submit\"
    value=\"Save\" onClick=\"saveprofile()\"";
217
218             if(REFLOW_STATUS == true){ printhtml += " disabled"; }
219
220              printhtml += "></div></div></div>";
221
222     return printhtml;
223 }
224
225 // Javascript on reboot :: Auto 5 seconds redirect homepage "/"
226 static const String jsreboot = "<script
    src=\"data:text/javascript; base64,cmVkaXJlY3RUaW1lT3V0ID0gIjUwMDAiO
    wpwYXRoID0gIi8iOwpzZXRUaW1lb3V0KCJsb2NhdGlvbi5ocmVmID0gcGF0aDsiLHJl
    ZGlyZWN0VGltZU91dCk7\"></script>";
227
228 String profiledata(int profilenum){
229     int ofs = 8*(profilenum-1);
230
231     String printjava = "<script>var temps =
    [25,"+String(profile_param[0+ofs])+","+String(profile_param[1+ofs])
    +","+String(profile_param[2+ofs])+","+String(profile_param[3+ofs])+
    "];"
232                         "var times =
    [0,"+String(profile_param[4+ofs])+","+String(profile_param[5+ofs])+
    ","+String(profile_param[6+ofs])+","+String(profile_param[7+ofs])+"
    ];</script>";
233
234     return printjava;
235 }
236
237 // PAGE : reflowdata //
238 String profileplotprint(){
239
240           String printjava = "";
241
242           for(int i = 0; i < plotleg; i++){
243             if(i != 0){
244               printjava += ",";
245             }
246             printjava +=
    String(plotdata[2*i])+","+String(plotdata[2*i+1]);
247           }
248
249           //printjava += "];";
250
251     return printjava;
252 }
```

B6. Código JavaScript decodificado

Para hacer el código fuente del archivo HTML.h más manejable se decidió codificar el código JavaScript en base64, por lo tanto en este apartado se encuentran los códigos JavaScript de la interfaz web decodificados.

Código fuente de la línea 111 del archivo HTML.h

```
001   // Update thermocouple temp
002   var temptimmer = setInterval(function() {
003     var xhttp = new XMLHttpRequest();
004     xhttp.onreadystatechange = function() {
005       if (this.readyState == 4 && this.status == 200) {
006         document.getElementById("temp").innerHTML =
007         this.responseText + " &deg;C";
008       }
009     };
010     xhttp.open("GET", "/temp", true);
011     xhttp.send();
012   }, 2551);
013
014   var graph = document.getElementById('graph');
015   var context = graph.getContext('2d');
016   var  width = graph.width = 656;
017   var height = graph.height = 425;
018   context.fillStyle = '#f6f6f6';
019   context.fillRect(0, 0, width, height);
020   context.fillStyle = "#000000";
021   context.font = "12px Arial";
022   var i = 25;
023   for(; i < height;) {
024     // Temp Value
025     context.fillText(i,1,height-i);
026     // Lineas Horizontales
027     context.beginPath();
028     context.moveTo(22, height-i-4);  // Xi,Yi
029     context.lineTo(width-30, height-i-4); // Xf,Yf
030     context.lineWidth = 0.25;
031     context.lineCap = 'round';
032     context.strokeStyle = '#3b3b3b';
033     context.stroke();
034     i = i+25;
035   }
036   context.font = "9px Arial";
037   context.fillText(String.fromCharCode(186)+"C",11,10);
038   context.fillStyle = "#000000";
039   context.font = "12px Arial";
040   context.textAlign = "center";
041   var i = 0;
042   for(; i < (width-30);) {
043     // Temp Value
044     context.fillText(i,25+i,height-10);
045
046     // Lineas Horizontales
```

```
047     context.beginPath();
048     context.moveTo(i+25, 20);  // Xi,Yi
049     context.lineTo(i+25, height-25); // Xf,Yf
050     context.lineWidth = 0.25;
051     context.lineCap = 'round';
052     context.strokeStyle = '#3b3b3b';
053     context.stroke();
054     i = i+30;
055   }
056   context.font = "9px Arial";
057   context.fillText("sec",width-12,height-10);
058   context.translate(25, -25);
059   // Print Reflow Profile
060   var prev_stat = temps[0],
061     prev_left = times[0];
062   for(stat in temps) {
063     the_stat = temps[stat];
064     the_time = times[stat]
065     context.beginPath();
066     context.moveTo(prev_left, height-prev_stat+21);  // Xi,Yi
067     context.lineTo(the_time, height-the_stat+21); // Xf,Yf
068     context.lineWidth = 2;
069     context.lineCap = 'round';
070     context.stroke();
071     prev_stat = the_stat;
072     prev_left = the_time;
073   }
074
075
076   var x = 1;
077   var plotdata = [0];
078   var plottimmer = setInterval(function() {
079
080         var xmlhttpplot=new XMLHttpRequest();
081         xmlhttpplot.onreadystatechange=function(){
082             if (xmlhttpplot.readyState==4 && xmlhttpplot.status==200){
083
084           plotdata = xmlhttpplot.responseText.split(",");
085
086            for(; x < (plotdata.length/2); x++) {
087
088              context.beginPath();
089              context.moveTo(plotdata[x*2+1],height-
    plotdata[x*2]+20);    // Xi,Yi
090              context.lineTo(plotdata[(x-1)*2+1], height-plotdata[(x-
    1)*2]+20);    // Xf,Yf
091              context.lineWidth = 1;
092              context.lineCap = 'round';
093              context.strokeStyle = '#ff0000';
094              context.stroke();
095
096           } // END FOR
097              }
098         }
099         xmlhttpplot.open("GET","/reflowdata",true);
100         xmlhttpplot.send();
101
102   }, 4111); // END TIMMER
```

Código fuente de la línea 135 del archivo HTML.h

```
1   function savewifi() {
2       var ssid = document.getElementById("ssid");
3     var pwd = document.getElementById("pwd");
4       var gourl = "http://" + window.location.host + "/wifi/" +
  ssid.value + "/" + pwd.value;
5       window.location = gourl;
6   }
```

Código fuente de la línea 143 del archivo HTML.h

```
01   function saveprofile(){
02     var name = document.getElementById("name");
03     var profilenum = document.getElementById("profilenum");
04     var temp1 = document.getElementById("temp1");
05     var temp2 = document.getElementById("temp2");
06     var temp3 = document.getElementById("temp3");
07     var temp4 = document.getElementById("temp4");
08     var time1 = document.getElementById("time1");
09     var time2 = document.getElementById("time2");
10     var time3 = document.getElementById("time3");
11     var time4 = document.getElementById("time4");
12     var gourl = "http://" + window.location.host + "/profile/" +
   profilenum.value + "/" + name.value + "/" + temp1.value + "/" +
   temp2.value + "/" + temp3.value + "/" + temp4.value + "/" +
   time1.value + "/" + time2.value + "/" + time3.value + "/" +
   time4.value;
13     window.location = gourl;
14   }
```